09/926320
PCT/JP00/02512

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

17.04.00

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: １９９９年 ４月２２日

出願番号
Application Number: 平成１１年特許願第１１５０４７号

出願人
Applicant (s): 関 一

２０００年 ５月１９日

特許庁長官
Commissioner,
Patent Office 近藤隆彦

出証番号 出証特２０００－３０３７０３５

【書類名】　特許願

【整理番号】　R002

【あて先】　特許庁長官殿

【国際特許分類】　G06F 15/00

【発明者】

【住所又は居所】　愛媛県松山市道後喜多町４番３８号

【氏名】　関　一

【特許出願人】

【識別番号】　598003070

【住所又は居所】　愛媛県松山市道後喜多町４番３８号

【氏名又は名称】　関　一

【手数料の表示】

【予納台帳番号】　057509

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　計算機システム

【特許請求の範囲】

【請求項１】

各々のエントリにデータが書き込まれるようになっている統合レジスタ･ファイル（６）と、

各々のエントリに該統合レジスタ･ファイル（６）のエントリのアドレスが書き込まれるようになっている前進ポインタ･スタック（３）と、

各々のエントリに個々の命令の内容が書き込まれるようになっているＦＩＦＯ(First In First Out) キューの構成となっている命令バッファ（５）と、

各々適当な数のリザベーション･ステーションを備える機能ユニット群と、

該統合レジスタ･ファイル（６）及び該機能ユニット群の間でデータを該統合レジスタ･ファイル（６）のエントリのアドレスと共に分配するようになっている共通データ･バス（８）とを具備し、

デコードされた命令においてオペランド･スタックに対するポップ操作が規定されている場合には、ポップすべき語数と同じ数だけ該統合レジスタ･ファイル（６）のエントリのアドレスを該前進ポインタ･スタック（３）からポップし、

デコードされた命令においてオペランド･スタックに対するプッシュ操作が規定されている場合には、プッシュすべき語数と同じ数だけ割り付けられていない該統合レジスタ･ファイル（６）のエントリを割り付け、該割り付けた該統合レジスタ･ファイル（６）のエントリのアドレスを該前進ポインタ･スタック（３）にプッシュし、

デコードされた命令の内容を、ポップ／プッシュ操作を伴う命令の場合にはポップ／プッシュされる該統合レジスタ･ファイル（６）のエントリのアドレスと共に、該命令バッファ（５）に書き込み、

命令の種類に応じて、必要な場合には、該命令バッファ（５）に書き込まれる命令の内容を、該命令を実行する機能ユニットの命令の内容を保持していないリザベーション･ステーションにも書き込み、

該前進ポインタ･スタック（３）からエントリ･アドレスがポップされる該統合

レジスタ･ファイル（６）のエントリの各々の内容が読み出され、データが既に書き込まれている場合には、エントリ･アドレスとデータが該共通データ･バス（８）に載せられるようになっており、

該リザベーション･ステーションの各々において、命令の内容を保持している場合、そこに書き込まれているソース･データを保持すべき該統合レジスタ･ファイル（６）のエントリのアドレスと該共通データ･バス（８）を通じて送られてくる該統合レジスタ･ファイル（６）のエントリのアドレスが比較され一致すればデータが取り込まれ、必要なソース･データが揃った後に該命令の実行が開始され、

該機能ユニット群の各々において、デコードの際該前進ポインタ･スタック（３）に該統合レジスタ･ファイル（６）のエントリのアドレスがプッシュされるような命令の実行の結果得られたデータを該プッシュされた該統合レジスタ･ファイル（６）の該エントリの該アドレスと共に該共通データ･バス（８）に載せ

該統合レジスタ･ファイル（６）においては、該共通データ･バス（８）を通じて送られてくる内容に基づきデータの書き込みを行うことによって、スタックマシンの機械語で記述されたプログラムを実行する計算機システム。

【請求項２】

各々のエントリに該統合レジスタ･ファイル（６）のエントリのアドレスが書き込まれるようになっている完了ポインタ･スタック（４）を具備し、

該命令バッファ（５）におけるキューの先頭のエントリに保持されている命令の完了が可能である、あるいはそうなると、該キューの該先頭のエントリの内容に基づき、該保持されている命令がデコードされた際の該前進ポインタ･スタック（３）の動作を再現すべく該完了ポインタ･スタック（４）を操作し、該キューから該先頭のエントリを除外し、

ポップ操作によって該完了ポインタ･スタック（４）におけるアドレスの保持が無くなった該統合レジスタ･ファイル（６）のエントリの割り付けを解除するようになっている請求項１記載の計算機システム。

【請求項３】

データ･キャッシュ（１１）とデータ･バッファ（１２）を具備し、

演算を実行するようになっている演算ユニット（８０、８１）と

該データ･キャッシュ（１１）及び該データ･バッファ（１２）にアクセスできるようになっているロード／ストア･ユニット（８３）とを該機能ユニット群に含む請求項２記載の計算機システム。

【請求項４】

割り付けられていない該統合レジスタ･ファイル（６）のエントリのアドレスを保持するフリー･リストを具備し、

初期状態においては、該統合レジスタ･ファイル（６）の全てのエントリのアドレスが該フリー･リストに登録されており、

該統合レジスタ･ファイル（６）のエントリを割り付ける必要がある場合に、該フリー･リストから割り付けられていない該統合レジスタ･ファイル（６）のエントリのアドレスを取り出し、

割り付けが解除された該統合レジスタ･ファイル（６）のエントリのアドレスを該フリー･リストに登録するようになっている請求項２記載の計算機システム。

【請求項５】

各々のエントリに該前進ポインタ･スタック（３）の内容が書き込まれるようになっている前進ポインタ･スタック履歴ファイル（３ａ）を具備し、

該統合レジスタ･ファイル（６）が各々のエントリに分岐タグも書き込まれるような構成となっており、

命令のデコードの際に、割り付けられる該統合レジスタ･ファイル（６）のエントリに分岐タグを書き込むようになっており、

条件分岐命令がデコードされるごとに、該前進ポインタ･スタック履歴ファイル（３ａ）の１エントリに該前進ポインタ･スタック（３）の内容を書き込み、分岐タグを変更して、分岐予測に基づく投機的実行を行い、

分岐予測が外れた場合には、その条件分岐命令以降にデコードされた命令を無効にし、該条件分岐命令以降にデコードされた命令に付した分岐タグが書き込まれている該統合レジスタ･ファイル（６）のエントリの割り付けを解除し、該条

件分岐命令がデコードされた際に書き込みの行われた該前進ポインタ･スタック履歴ファイル（３ａ）のエントリの内容を、該前進ポインタ･スタック（３）にコピーして、正しい位置の命令から処理を再開することによって、

分岐予測に基づく投機的実行を行うようになっている請求項２記載の計算機システム。

【請求項６】

該前進ポインタ･スタック（３）と該完了ポインタ･スタック（４）が循環型のバッファの構成となっており、

該前進ポインタ･スタック（３）及び該完了ポインタ･スタック（４）で、該統合レジスタ･ファイル（６）のエントリのアドレスを保持する最下位のエントリの内容が一致する場合には、該前進ポインタ･スタック（３）及び該完了ポインタ･スタック（４）において該最下位のエントリにおける該統合レジスタ･ファイル（６）のエントリのアドレスの保持を解除し、該一致する内容で示される該統合レジスタ･ファイル（６）のエントリに書き込まれているデータを該データ･バッファ（１２）にストア（Spill）することができるようになっており、

該データ･バッファ（１２）に最後にストア（Spill）したデータに対し、割り付けられていない該統合レジスタ･ファイル（６）の１エントリを割り付け、該データを書き込み、該前進ポインタ･スタック（３）及び該完了ポインタ･スタック（４）において、該統合レジスタ･ファイル（６）のエントリのアドレスを保持する最下位のエントリの１つ下のエントリに該データが書き込まれる該統合レジスタ･ファイル（６）のエントリのアドレスを保持させることによって、該最後にストア（Spill）したデータの該統合レジスタ･ファイル（６）へのロード（Fill）を行えるようになっている請求項３記載の計算機システム。

【請求項７】

該フリー･リストがＦＩＦＯキューの構成となっており、

同時に複数の命令をデコードし、該前進ポインタ･スタック（３）の操作、該統合レジスタ･ファイル（６）のエントリの割り付け及び該命令バッファ（５）の連続する複数のエントリへの命令の内容の書き込みを行う機能と、

同時に該命令バッファ（５）の連続する複数のエントリに書き込まれている内

容に基づき、該完了ポインタ･スタック（４）の操作及び該統合レジスタ･ファイル（６）のエントリの割り付けの解除を行う機能を有する請求項４記載の計算機システム。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】

本発明は、スタックマシンの機械語で記述されたプログラムを高速で処理する新規な構成の計算機システムに関するものである。

【０００２】

【従来の技術】

従来、スタックマシンにおいては、命令の実行は、基本的にプログラム上の順序通り（in-order）に行われるものであった。すなわち、スタックマシンにおける演算命令は、オペランド･スタックからソース･データをポップし、演算を実行し、その演算結果をオペランド･スタックにプッシュするというような動作を指示するものであるが、このような命令の連鎖として書かれたプログラムを逐次的に実行するのである。

【０００３】

【発明が解決しようとする課題】

上述したような従来のスタックマシンにおいては、命令をプログラム上の順序通り（in-order）に実行するので、制御構造が単純なもので済むという利点があるが、処理速度が制約を受けるという問題点があった。

【０００４】

そこで、スタックマシンの機械語で記述されたプログラムをout-of-orderで処理するような計算機方式が考案された。例えば、日本特公平２－２６００８２号、米国特許第５５２２０５１号や、米国特許第５３３３３２０号及び米国特許第５７６５０１４号におけるプロセッサ要素がある。これらの明細書に示されるプロセッサは、処理性能の向上という点で十分ではない上に、正確な例外処理を保証する上で問題があった。

【０００５】

本発明は、上記問題点を解決するため創案されたものであり、正確な例外処理を保証しつつ、スタックマシンの機械語で記述されたプログラムをout-of-orderでより効率的に処理する計算機システムを提供することを目的としている。

【０００６】

【課題を解決するための手段】

本発明による計算機システムは、データ･キャッシュと、データ･バッファと、各々のエントリにデータが書き込まれるようになっている統合レジスタ･ファイルと、各々のエントリに統合レジスタ･ファイルのエントリのアドレスが書き込まれるようになっている前進ポインタ･スタック及び完了ポインタ･スタックと、各々のエントリに個々の命令の内容が書き込まれるようになっているＦＩＦＯキューの構成となっている命令バッファと、演算を実行するようになっている演算ユニットとデータ･バッファ及びデータ･キャッシュにアクセスできるようになっているロード／ストア･ユニットを含む機能ユニット群と、統合レジスタ･ファイル及び機能ユニット群の間でデータを統合レジスタ･ファイルのエントリのアドレスと共に分配するようになっている共通データ･バスとを具備する。上記機能ユニットの各々は、適当な数のリザベーション･ステーションを備える。

【０００７】

従来のスタックマシンにおいて、スタックが ....., word1, word2, word3, word4（右端がスタックトップ）となっている状態は、本発明による計算機システムにおいて、ポインタ･スタックが .....,〈a〉,〈b〉,〈c〉,〈d〉（右端がスタックトップ）で、エントリ･アドレスが〈a〉,〈b〉,〈c〉及び〈d〉である統合レジスタ･ファイルの各エントリに、それぞれ word1, word2, word3 及び word4 が保持されている状態に対応する。

【０００８】

本発明の計算機システムにおいては、命令がデコードされるごとに、命令の内容に応じて前進ポインタ･スタック及び統合レジスタ･ファイルを操作すると共に、命令の内容を命令バッファ及び、必要な場合には、適切な機能ユニットの空いているリザベーション･ステーションに書き込むようになっている。この際、命令に規定されているオペランド･スタックに対するスタック操作が、前進ポイン

タ･スタックに対して同様に適用される。ここで、１語のデータのオペランド･スタックへのプッシュ操作を、本発明の計算機システムにおいてエミュレートするには、そのデータを保持すべく統合レジスタ･ファイルの空いている１エントリを割り付け、そのエントリのアドレスを前進ポインタ･スタックにプッシュすればよい。

【０００９】

即ち、デコードされた命令においてオペランド･スタックに対するポップ操作が規定されている場合には、ポップすべき語数と同じ数だけ統合レジスタ･ファイルのエントリのアドレスを前進ポインタ･スタックからポップする。デコードされた命令においてオペランド･スタックに対するプッシュ操作が規定されている場合には、プッシュすべき語数と同じ数だけ統合レジスタ･ファイルの空いているエントリを割り付け、上記割り付けた統合レジスタ･ファイルのエントリのアドレスを前進ポインタ･スタックにプッシュする。さらに、デコードされた命令の内容を、ポップ／プッシュ操作を伴う命令の場合にはポップ／プッシュされる統合レジスタ･ファイルのエントリのアドレスと共に、命令バッファに書き込むようになっている。機能ユニットによる実行の必要な命令の場合には、命令バッファに書き込まれる命令の内容を、適切な機能ユニットの空いているリザベーション･ステーションにも書き込むようになっている。

【００１０】

前進ポインタ･スタックからエントリ･アドレスがポップされる統合レジスタ･ファイルのエントリの各々の内容が読み出され、データが既に書き込まれている場合には、後で、エントリ･アドレスとデータが共通データ･バスに載せられるようになっている。

【００１１】

リザベーション･ステーションに書き込まれた命令に関して、原則として次のような動作が順次行われる。各々のリザベーション･ステーションで、そこに書き込まれているソース･データを保持すべき統合レジスタ･ファイルのエントリのアドレスと共通データ･バスを通じて送られてくるものが比較され、一致すればデータが取り込まれる。必要なソース･データが揃った後に、その命令の実行が

開始される。デコードの際前進ポインタ･スタックに統合レジスタ･ファイルのエントリのアドレスがプッシュされるような命令の場合、機能ユニットでの実行の結果得られたデータを該プッシュされた統合レジスタ･ファイルのエントリのアドレスと共に共通データ･バスに載せる。統合レジスタ･ファイルにおいては、共通データ･バスを通じて送られてくる内容に基づきデータの書き込みを行う。

【００１２】

命令バッファにおけるキューの先頭のエントリに保持されている命令の完了が可能である、あるいはそうなると、そのキューの先頭のエントリの内容に基づき、保持されている命令がデコードされた際の前進ポインタ･スタックの動作を再現すべく完了ポインタ･スタックを操作し、キューからその先頭のエントリを除外し、ポップ操作によって完了ポインタ･スタックにおけるアドレスの保持が無くなった統合レジスタ･ファイルのエントリの割り付けを解除するようになっている。

【００１３】

【実施例】

以下に、本発明による計算機システムの一実施例について、図面を参照しながら説明する。なお、以下に述べる本発明による計算機システムの実施例は、Java Virtual Machine（Java VM）で規定されるスタックマシンの基本的な命令をハードウエアで実行するものである。すなわち、データ語長を32ビットとして、これを単位にロード／ストア及び算術論理演算等の演算を行う。従って、例えば、倍長語の間での算術演算は、２語づつ合せて４語のソース･データをもとに２語の演算結果を生ずる。

【００１４】

従来のスタックマシンにおける、語の単位でデータがプッシュ／ポップされるようになっているスタックは、後述するポインタ･スタックと区別するために、以降では、ワード･スタックと呼ぶことにする。

【００１５】

Java VM においては、ワード･スタックには、メソッドが呼び出されるごとにフレームが積まれる。各フレームで、下部はローカル変数やパラメータの格納域

、上部はオペランド･スタックとなっている。

【００１６】

Java VM にはもともとハードウェアで実行することを想定していない複雑な命令が含まれるが、以下に述べる本発明による計算機システムの実施例は、次のような基本的な命令をハードウェアで実行するものとする。

（ａ）即値データのオペランド･スタックへのプッシュ命令

bipush, sipush, aconst_null, iconst_ml, iconst_<i>, fconst_<f>, lconst_<l>, dconst_<d>

（ｂ）変数データのオペランド･スタックへのロード命令

ldc1, ldc2, iload, iload_<n>, fload, fload_<n>, aload, aload_<n>, ldc2w, lload, lload_<n>, dload, dload_<n>, iaload, laload, faload, daload, aaload, baload, caload, saload

（ｃ）オペランド･スタック上のデータの変数へのストア命令

istore, istore_<n>, fstore, fstore_<n>, astore, astore_<n>, lstore, lstore_<n>, dstore, dstore_<n>, iastore, lastore, fastore, dastore, aastore, bastore, castore, sastore

（ｄ）演算命令

（ｄ－１）算術演算命令

iadd, ladd, fadd, dadd, isub, lsub, fsub, dsub, imul, lmul, fmul, dmul, idiv, ldiv, fdiv, ddiv, irem, lrem, frem, drem, ineg, lneg, fneg, dneg

（ｄ－２）論理演算命令

ishl, ishr, iushr, lshl, lshr, lushr, iand, land, ior, lor, ixor, lxor

（ｄ－３）変換演算命令

i2l, i2f, i2d, l2i, l2f, l2d, f2i, f2l, f2d, d2i, d2l, d2f, int2byte, int2char, int2short

（ｄ－４）比較演算命令

lcmp, fcmpl, fcmpg, dcmpl, dcmpg

（ｅ）オペランド･スタックの操作命令

pop, pop2, dup, dup2, dup_x1, dup2_x1, dup_x2, dup2_x2, swap

（ｆ）分岐命令

ifeq, ifnull, iflt, ifle, ifne, ifnonnull, ifgt, ifge, if_icmpeq, if_icmpne, if_icmplt, if_icmpgt, if_icmple, if_icmpge, goto, goto_w

〔００１７〕

以降、特にことわらない限り、「命令」とは上に挙げた命令のいずれかを意味するものとする。

〔００１８〕

図１は計算機システムのブロック図であって、１０は命令キャッシュ、１１はデータ･キャッシュ、１２はデータ･バッファ、２０は命令フェッチ･ユニット、２１は命令デコード･発行ユニット、３は前進ポインタ･スタック、３ａは前進ポインタ･スタック履歴ファイル、４は完了ポインタ･スタック、５は命令バッファ、６は統合レジスタ･ファイル、７はフリー･リスト、８は共通データ･バス、８０及び８１は各々演算ユニット０及び１、８２は分岐ユニット、８３はロード／ストア･ユニット、８０１，８０２，８１１，８１２，８２１，８２２，８３１及び８３２はリザベーション･ステーションを表している。

〔００１９〕

次に、本発明実施例の計算機システムの各構成要素ごとにその詳細な構成を説明する。

〔００２０〕

（Ａ）命令フェッチ･ユニット

命令フェッチ･ユニットは、図示してないプログラムカウンタ（pcレジスタ）を具備しており、命令キャッシュから命令をフェッチし、命令デコード･発行ユニットに渡すようになっている。分岐の予測や分岐の実行も担う。

〔００２１〕

（Ｂ）命令デコード･発行ユニット

命令デコード･発行ユニットは、命令フェッチ･ユニットから渡された命令のデコードを行い、プログラムに含まれる命令がout-of-orderで実行されるように、後述する前進ポインタ･スタック、命令バッファ及び統合レジスタ･ファイル等を設定するための各種信号を発生するようになっている。

【００２２】

（Ｃ）ポインタ･スタック

ポインタ･スタックは、各々のエントリに統合レジスタ･ファイルのエントリのアドレスが書き込まれるようになっているレジスタ･ファイルで構成されている。

【００２３】

従来のスタックマシンにおいて、ワード･スタックが ....., word1, word2, word3, word4（右端がスタックトップ）となっている状態は、本発明による計算機システムにおいて、ポインタ･スタックが .....,〈ａ〉,〈ｂ〉,〈ｃ〉,〈ｄ〉（右端がスタックトップ）で、エントリ･アドレスが〈ａ〉,〈ｂ〉,〈ｃ〉及び〈ｄ〉である統合レジスタ･ファイルの各エントリに、それぞれ word1, word2, word3 及び word4 が保持されている状態に対応する。

【００２４】

本発明の計算機システムは、前進ポインタ･スタック（ＡＰＳ；Advanced Pointer Stack）と完了ポインタ･スタック（ＣＰＳ；Completed Pointer Stack）の２つのポインタ･スタックを具備する。

【００２５】

本発明の計算機システムにおいては、命令がデコードされるごとに、命令の内容に応じて前進ポインタ･スタック（以下ではＡＰＳと記す）及び統合レジスタ･ファイルを操作すると共に、命令の内容を命令バッファ及び、必要な場合には、適切な機能ユニットの空いているリザベーション･ステーションに書き込むことにより、プログラムに含まれる命令がout-of-orderで実行されるべく設定される。すなわち、前進ポインタ･スタックはデコード･発行済みの全ての命令によるスタック操作を反映している。

【００２６】

他方、完了ポインタ･スタック（以下ではＣＰＳと記す）は、プログラム上の順番で完了済みの全ての命令によるスタック操作を反映するものである。本発明の計算機システムはデータ駆動の原理に基づく out-of-order 実行を可能とするものであるが、完了ポインタ･スタックは、正確な例外処理を保証するため、in-

orderで完了済の全ての命令に基づく状態を構成するために存在するものである。

【００２７】

本発明実施例の計算機システムにおいては、ポインタ･スタック及び統合レジスタ･ファイルによって、ワード･スタックの一番上に積まれたフレームの上部のオペランド･スタックの部分のみが保持されるようになっている。ワード･スタックの残りの部分はデータ･バッファ及びデータ･キャッシュに格納されるようになっている。また、オペランド･スタックが成長して、ポインタ･スタック及び統合レジスタ･ファイルで保持しきれなくなると、後述するように、オペランド･スタックの下部がデータ･バッファにSpillされるようになっている。

【００２８】

各ポインタ･スタックは循環型のバッファの構成となっており、プッシュ･ポインタとボトム･ポインタと呼ぶ２つのレジスタが各々存在する。プッシュ･ポインタは、統合レジスタ･ファイルのエントリのアドレスを保持する最上位のエントリの１つ上を示す。ボトム･ポインタは、統合レジスタ･ファイルのエントリのアドレスを保持する最下位のエントリを示す。ボトム･ポインタの値からプッシュ･ポインタの値を引くことで、ポインタ･スタックに何エントリの空きがあるかがわかる。初期状態においては、プッシュ･ポインタ及びボトム･ポインタの各々の値は共に０となっている。

【００２９】

図２は、本実施例の計算機システムにおける、各ポインタ･スタックと各プッシュ･ポインタ及びボトム･ポインタの関係を示す説明図である。２つのポインタ･スタックＡＰＳ３及びＣＰＳ４は同数のエントリを有し、各ポインタ･スタックで各々のエントリに下から順に０、１、２、･･･ とアドレスが付けられているものとする。縦線が施されているエントリは統合レジスタ･ファイルのエントリのアドレスを保持しているものとする。図２に示すように、プッシュ･ポインタは、ＡＰＳ及びＣＰＳの各々に対して設けられており、それぞれPP_OF_APS及びPP_OF_CPSと名付けている。他方、ボトム･ポインタは１つだけ設けられており、これがＡＰＳ及びＣＰＳで共用される。これをBP_OF_PSと名付けている。

【００３０】

ＡＰＳとＣＰＳの間には、エントリの数だけ比較回路が設けられており、ＡＰＳ及びＣＰＳの同じエントリ･アドレスにある（図２において水平に並ぶ）エントリの間でその内容が比較されるようになっている。

【００３１】

命令のデコード･発行の際、命令に規定されているオペランド･スタックに対する１語分のプッシュ操作に対応して、割り付けられる統合レジスタ･ファイルの１エントリのアドレスをＡＰＳのPP_OF_APSで示されるエントリに書き込み、PP_OF_APSの値に１を加えるようになっている。逆に、命令に規定されているオペランド･スタックに対する１語分のポップ操作に対応して、PP_OF_APSの値から１を引くようになっている。命令の完了の際のＣＰＳとPP_OF_CPSの動作に関しても同様である。

【００３２】

BP_OF_PSで示されるエントリの内容がＡＰＳとＣＰＳで一致する場合には、その一致する内容で示される統合レジスタ･ファイルのエントリに書き込まれている１語分のデータをデータ･バッファにSpillすることができる。その際、BP_OF_PSの値に１を加えるようになっている。逆に、データ･バッファから統合レジスタ･ファイルに１語分のデータをFillするには、データ･バッファからFillすべき１語分のデータを取り出し、それに空いている統合レジスタ･ファイルの１エントリを割り付けてそのデータを書き込み、その統合レジスタ･ファイルのエントリのアドレスをＡＰＳ及びＣＰＳのBP_OF_PSで示されるエントリの１つ下に各々書き込み、BP_OF_PSの値から１を引くようになっている。

【００３３】

本実施例の計算機システムは、分岐予測に基づく投機的実行を可能にするために、前進ポインタ･スタック履歴ファイル（以下では「ＡＰＳ履歴ファイル」と記す）を具備する。ＡＰＳ履歴ファイルの各々のエントリには、ＡＰＳの全エントリ及びPP_OF_APSの内容が書き込めるようになっている。

【００３４】

（Ｄ）統合レジスタ･ファイル（ＣＲＦ；Consolidated Resister File）

統合レジスタ･ファイル（以下ではＣＲＦと記す）は、従来のスタックマシンにおけるオペランド･スタックの内容を、順序不同で保持するものである。

【００３５】

図３は、本実施例の計算機システムにおける、ＣＲＦ６の各々のエントリ６(i)の詳細な構成を示す説明図である。ここで、ｉはエントリのアドレスである。ＣＲＦ６の各々のエントリ６(i)はデータ･フィールド６１(i)、書込み完了フラグ（ＷＣＦ，Write Completion Flag）フィールド６２(i)、カラー（Ｃ，Colour）フィールド６３(i)及びビジービット（ＢＢ）フィールド６４(i)から成っている。

【００３６】

実際のＣＲＦのハードウェア上の構成は、上述の各フィールド別に設けられたレジスタ･ファイルの集合体である。

【００３７】

ＣＲＦの各々のエントリのデータ･フィールドは、１語分のデータが書き込まれる構成となっている。

【００３８】

ＣＲＦの各々のエントリにおいて、ＷＣＦフィールドは、データ･フィールドにデータの書き込みが完了していれば１、完了していなければ０が書き込まれているようになっている。

【００３９】

ＣＲＦの各々のエントリにおいて、Ｃフィールドは、そのＣＲＦのエントリが、命令に含まれるプッシュ操作に対応して割り付けられたものであるのか、データ･バッファからのFillの際に割り付けられたものであるのかの区別、前者の場合にはさらに分岐タグが書き込まれるようになっている。本実施例においては、後述するように、分岐タグはＡＰＳ履歴ファイルのエントリのアドレスと一定の関係にある。

【００４０】

ＣＲＦの各々のエントリにおいて、ＢＢフィールドは、そのＣＲＦのエントリがデータを保持すべく割り付けられている状態であれば１、割り付けられていな

い（空いている）状態であれば０が書き込まれているようになっている。

【００４１】

（Ｅ）フリー･リスト（ＦＬ）

フリー･リスト（以下ではＦＬと記す）は、フリーな、即ち、空いている／割り付けられていない（ＢＢフィールドが０である）ＣＲＦのエントリのアドレスを保持するもので、本実施例においては、循環型のＦＩＦＯキューの構成となっている。

【００４２】

初期状態においては、ＣＲＦの全てのエントリのアドレスがＦＬに登録されている。空いているＣＲＦのエントリを割り付ける必要がある場合に、ＦＬからフリーなＣＲＦのエントリのアドレスが取り出される。逆に、ＣＲＦのあるエントリの割り付けが解除されれば、そのエントリのアドレスがＦＬに登録されるようになっている。

【００４３】

（Ｆ）命令バッファ（ＩＢ；Instruction Buffer）

命令バッファ（以下ではＩＢと記す）は、未完了の発行済命令を保持するバッファであり、循環型のＦＩＦＯキューの構成となっている。

【００４４】

図４は、ＩＢの構成を示す説明図である。図４において、ＩＢ５の各々のエントリは下から順に０、１、２、･･･とアドレスが付けられているものとし、縦線が施されているＩＢ５のエントリは、未完了の発行済命令を保持しているものとする。ＩＢは、ヘッダ･ポインタとトレイル･ポインタと名付けた２つのレジスタを具備する。ヘッダ･ポインタはキューの先頭のエントリを、トレイル･ポインタはキューの末尾のエントリの１つ後を示す。１サイクル当たり１命令までしか発行／完了を行わないものとすれば、トレイル･ポインタは次に発行される命令の内容を書き込むべきエントリを示し、ヘッダ･ポインタは次に完了されるべき命令の内容が書き込まれているエントリを示す。ヘッダ･ポインタの値からトレイル･ポインタの値を引くことで、ＩＢに何エントリの空きがあるかがわかる。初期状態においては、ヘッダ･ポインタ及びトレイル･ポインタの値は共に０となっ

ている。

【００４５】

図５は、本実施例の計算機システムにおける、ＩＢ５の各々のエントリ５(i)の詳細な構成を示す説明図である。ここで、ｉはエントリのアドレスである。ＩＢ５の各々のエントリ５(i)はオペレーション･フィールド５０(i)、オペランド･フィールド５１(i)、第１ソース･フィールド５２(i)、第２ソース･フィールド５３(i)、第３ソース･フィールド５４(i)、第４ソース･フィールド５５(i)、第１デスティネーション･フィールド５６(i)、第２デスティネーション･フィールド５７(i)、分岐タグ（ＢＴ）フィールド５８(i)、及び実行状態（Ｓ；State）フィールド５９(i)から成っている。

【００４６】

ＩＢの各々のエントリのオペレーション･フィールドはオペレーション･コードが書き込まれる構成となっている。

【００４７】

ＩＢの各々のエントリのオペランド･フィールドは、オペレーション･コードに続いてオペランドが示されるような命令の場合に、このオペランドが書き込まれるようになっている。

【００４８】

ＩＢの各々のエントリの第１～第４ソース･フィールドの各々は、ソース･データを保持すべく割り付けられているＣＲＦのエントリのアドレスが書き込まれるようになっている。ポップ操作を含む命令がデコードされた場合には、ポップすべき語数と同じ数だけＡＰＳからポップされるＣＲＦのエントリのアドレスが、その順で第１～第４ソース･フィールドに書き込まれるようになっている。

【００４９】

ＩＢの各々のエントリの第１～第２デスティネーション･フィールドの各々は、命令のデコード･発行に伴い、新たに割り付けられるＣＲＦのエントリのアドレスが書き込まれるようになっている。プッシュ操作を含む命令がデコードされた場合には、プッシュすべき語数と同じ数だけＡＰＳにプッシュされるＣＲＦのエントリのアドレスが、その順で第１～第２デスティネーション･フィールドに

書き込まれるようになっている。

【００５０】

ＩＢの各々のエントリのＢＴフィールドは、分岐予測に基づく投機的実行に係るもので、本実施例においては、後述するように、ＢＴフィールドに書き込まれる分岐タグはＡＰＳ履歴ファイルのエントリのアドレスと一定の関係にある。

【００５１】

ＩＢの各々のエントリにおいて、Ｓフィールドは、そのエントリに書き込まれている命令の実行状態に応じて、未実行、実行済み、正常終了、例外事象発生等の情報が書き込まれているようになっている。

【００５２】

（Ｇ）共通データ･バス（ＣＤＢ；Common Data Bus）

共通データ･バス（以下ではＣＤＢと記す）は、後述する機能ユニット群及びＣＲＦの間で、データをＣＲＦのエントリのアドレスと共に分配する通信チャネルである。ＣＤＢは、十分なデータ通信バンド幅が確保できるように、多重化されている。

【００５３】

（Ｈ）機能ユニット

本実施例の計算機システムは、演算ユニット０及び１、分岐ユニット及びロード／ストア･ユニットの４つの機能ユニットを具備する。本実施例においては、各機能ユニットは、基本的に、２つのリザベーション･ステーションと割り当てられた命令を処理する実行部で構成される。リザベーション･ステーション（以下ではＲＳと記す）は、命令の内容を一時的に保持するバッファであるが、本実施例の計算機システムにおいては、命令のデコードの際に、同じ命令の内容が書き込まれるＩＢのエントリのアドレスも書き込まれるような構成となっている。

【００５４】

命令のデコードの際に、命令の種類に応じて、必要な場合に、適切な機能ユニットの空いているＲＳに書き込みが行われるようになっている。

【００５５】

各機能ユニットの各々のＲＳで、そこに書き込まれているソース･データを保

持すべきＣＲＦのエントリのアドレスとＣＤＢで送られてくるものが比較され、一致すればデータが取り込まれるようになっている。

【００５６】

命令の内容を保持しているＲＳにおいて、必要なソース･データが揃い、機能ユニットの実行部が利用可能であれば、当該ＲＳの内容は実行部に渡され、その実行が開始されるようになっている。

【００５７】

（Ｈ－１）演算ユニット

本実施例の計算機システムは、演算ユニット０及び演算ユニット１を具備しており、その各々の実行部は算術論理演算、データ･タイプの変換演算、比較演算等の演算命令を実行する機能を有し、互いに独立に並行して動作することができるようになっている。

【００５８】

本発明の計算機システムにおいては、各々の演算ユニットの実行部をパイプライン化したり、より多くの演算ユニットを具備したり、演算ユニットごとに実行する演算の種類を特定した構成とすることも可能である。

【００５９】

（Ｈ－２）分岐ユニット

分岐ユニットの実行部は、条件分岐命令を処理し、分岐の有無を確定して、分岐先アドレスと共に、命令フェッチ･ユニットに通知する機能を有する。

【００６０】

（Ｈ－３）ロード／ストア･ユニット（ＬＳＵ；Load/Store Unit）及びデータ･バッファ

ロード／ストア･ユニット（以下ではＬＳＵと記す）の実行部は、アドレス計算を行う機能を有し、データ･バッファ及びデータ･キャッシュにアクセスすることができるようになっている。

【００６１】

データ･バッファは、各々のエントリに１語のデータが書き込まれるようになっている循環型のバッファである。本発明の計算機システムにおいては、ワード

・スタックの最上位の部分がポインタ・スタックとＣＲＦによって保持されるが、その下の部分がデータ・バッファ、さらにその下の部分がデータ・キャッシュに格納されるようになっている。ＬＳＵはデータ・バッファに高速にアクセスできるので、アクセスすべき変数データがデータ・バッファに保持されている割合が大きいほど、より効率的な計算が可能となる。また、データ・バッファに適当な語数のデータを溜めておくようにすることによって、後述するＣＲＦ－データ・バッファ－データ・キャッシュの間のSpill/Fillの動作を効率的に行うことができる。

【００６２】

ＬＳＵは、最初のローカル変数へのポインタを保持する図示してないレジスタ（varsレジスタ）を具備する。本実施例の計算機システムにおいては、最初のローカル変数の格納域はデータ・バッファあるいはデータ・キャッシュにあるが、varsレジスタには、データ・キャッシュにおける相当するアドレス値が書き込まれているようになっている。すなわち、全てあるいは一部のローカル変数のデータが実際にはデータ・バッファに保持されていても、各々のローカル変数に、全てのローカル変数をデータ・キャッシュにSpillしたと仮定した場合のデータ・キャッシュにおけるアドレス値を対応させることができるので、ロード／ストア命令の処理において、ＬＳＵはvarsレジスタの値を用いてアドレス計算を行い、対象となるローカル変数の格納域がデータ・バッファかデータ・キャッシュかを判定し、その格納域にアクセスする。

【００６３】

ＬＳＵは、先行命令が全て完了するまでストア命令をプログラム上の順番で保持する、図示してないストア・バッファを具備する。即ち、ストア命令は全ての先行命令が完了してから実行されるようになっている。ストア・バッファは連想機能を備えており、ＬＳＵは先行ストア命令に対する依存性の検証を行い、ロード命令の実行をout-of-orderで行うことができるようになっている。

【００６４】

即ち、ロード・アドレスが先行ストア命令のストア・アドレスに一致するか、あるいは、先行ストア命令のストア・アドレスが未計算の場合（この場合、依存関

係の検証はできないので、依存関係は存在するとみなす）、当該ロード命令は先行ストア命令に対して依存関係を持つことになる。依存関係が全く存在しない場合、データ･バッファ／データ･キャッシュから直ちにデータをロードする。ロード命令が先行ストア命令に対して依存関係にあると、データ･バッファ／データ･キャッシュは正しい値を持っていないので、データ･バッファ／データ･キャッシュからデータをロードすることはできない。ロード･アドレスが先行ストア命令のストア･アドレスと一致し、ストア･データが有効であれば、そのストア命令の完了を待たずに、ストア･バッファから直接データをロードする。

〔００６５〕

ＬＳＵは、プログラム中に示されるロード／ストア命令を実行すると共に、オーバーフロー／アンダーフローの回避のため、あるいは、メソッドの呼び出し／実行終了に伴いワード･スタックの最上位においてフレームが生成／破棄されるのに対応して、ＣＲＦに保持されているスタックの最下位にあたるデータをデータ･バッファとの間で自動的にSpill/Fillするようになっている。（ちなみに、メソッドの呼び出しにおいては、varsレジスタの値を変更した上で、スタック･トップからのストア命令も併用するのが望ましい。）

〔００６６〕

１語分のデータをＣＲＦからデータ･バッファにSpillするには、ＡＰＳとＣＰＳで、（BP_OF_PSで示される）ＣＲＦのエントリのアドレスを保持する最下位のエントリの内容が一致していなくてはならない（そうでない場合は一致するまで待つ）。その場合、その一致する内容で示されるＣＲＦのエントリに書き込まれている１語分のデータをデータ･バッファにSpillすることができる。その際、BP_OF_PSの値に１を加え、上記ＣＲＦのエントリのＢＢフィールドを０に変更し、そのエントリのアドレスをＦＬに登録する。

〔００６７〕

逆に、データ･バッファからＣＲＦに１語分のデータをFillするには、データ･バッファからFillすべき１語分のデータを取り出し、それに空いているＣＲＦの１エントリを割り付け、そのデータ･フィールドに書き込む。ＷＣＦ、ＢＢの各フィールドは１とする。さらに、その割り付けられたＣＲＦのエントリのアドレ

スを、ＡＰＳ及びＣＰＳの（BP_OF_PSで示される）ＣＲＦのエントリのアドレスを保持する最下位のエントリの１つ下に各々書き込み、BP_OF_PSの値から１を引く。

【００６８】

データ･バッファとデータ･キャッシュの間でも、データ･バッファの空きに応じて適宜Spill/Fillの動作が行われるようになっている。

【００６９】

ＣＲＦ－データ･バッファ－データ･キャッシュの間で一度に複数語のデータをSpill/Fillできるようにするには、ＡＰＳとＣＰＳの２つのポインタ･スタック、データ･バッファ及びデータ･キャッシュをインタリーブ分割して、対応する分割部分間で上述と同様な動作を行うような構成とすればよい。この場合、Spill/Fillのために、ＡＰＳとＣＰＳの２つのポインタ･スタック、データ･バッファ及びデータ･キャッシュにおいて、バンク毎に１つのread/writeポート、さらに、ＣＲＦにおいてインタリーブ分割の数だけのread/writeポートが必要となる。

【００７０】

ついで、本発明実施例の計算機システムの動作を説明する。

【００７１】

本実施例の計算機システムは命令を、①命令フェッチ、②命令デコード･発行、③実行、④完了の４ステージで処理する。当分の間、説明を簡単にするため、１サイクルで１つの命令をデコード･発行／完了できるものとして、以下に各ステージごとに動作内容を説明する。

【００７２】

① 命令フェッチ･ステージ

このステージでは、命令フェッチ･ユニットが命令キャッシュから命令を取り出すと共に、次にフェッチする命令のアドレスを決定する。次に命令をフェッチするのは通常次アドレス値からであるが、フェッチした命令が無条件分岐命令であるか、条件分岐命令で分岐すると予測した場合、分岐予測が外れた場合、あるいは例外が発生した場合には、フェッチするアドレス値を変更する。

【００７３】

② 命令デコード･発行ステージ

このステージでは、命令をデコードして、命令の内容に応じて前進ポインタ･スタック（ＡＰＳ）及び統合レジスタ･ファイル（ＣＲＦ）を操作すると共に、命令の内容を命令バッファ（ＩＢ）及び、必要な場合には、適切な機能ユニットの空いているＲＳに書き込むことにより、プログラムに含まれる命令がout-of-orderで実行されるべく設定する。以下に、設定動作を詳細に説明する。

【００７４】

本発明の計算機システムにおいては、従来のスタック･マシンにおけるワード･スタックのスタックトップ近傍がポインタ･スタックとＣＲＦによって再現されるが、命令に規定されているオペランド･スタックに対するスタック操作が、ＡＰＳに対して同様に適用される。ここで、１語のデータのオペランド･スタックへのプッシュ操作をエミュレートするには、そのデータを保持すべく空いているＣＲＦの１エントリを割り付け、そのエントリのアドレスをＡＰＳにプッシュすればよい。

【００７５】

即ち、デコードされた命令においてオペランド･スタックに対するポップ操作が規定されている場合には、ポップすべき語数と同じ数だけＣＲＦのエントリのアドレスをＡＰＳからポップする。デコードされた命令においてオペランド･スタックに対するプッシュ操作が規定されている場合には、プッシュすべき語数と同じ数だけ空いているＣＲＦのエントリを割り付け、上記割り付けたＣＲＦのエントリのアドレスをＡＰＳにプッシュする。

【００７６】

オペランド･スタックの操作命令（Java VM における pop，pop2，dup，dup2，dup_x1，dup2_x1，dup_x2，dup2_x2，swap）の場合、基本的には、オペランド･スタックに対して行うべき操作をＡＰＳに対して同様に行えばよい。本実施例においては、スタック上でコピーを作成するようなオペランド･スタックの操作命令（ Java VM における dup，dup2，dup_x1，dup2_x1，dup_x2，dup2_x2 ）の場合には、コピー･データを保持すべく空いているＣＲＦのエントリを割り付け、そのエントリのアドレスをＡＰＳの適切なエントリに書き込むようになっている

【００７７】

命令のデコード･発行に伴い新たに割り付けられるＣＲＦのエントリにおいては、ＢＢフィールドに１を立て、Ｃフィールドには命令デコード･発行ユニットから送られてくる分岐タグを書き込む。即値データのプッシュ命令の場合には、データがすでに得られているので、データ･フィールドにそのデータを書き込み、ＷＣＦフィールドに１を立てる。それ以外の場合には、データはデコード･発行の時点では得られていないので、ＷＣＦフィールドを０としておく。

【００７８】

デコードされた命令の内容をプログラム上の順番でＩＢに保持しておくために、その命令の内容をＩＢのトレイル･ポインタで示されるエントリに書き込み、トレイル･ポインタの値に１を加える。すなわち、オペレーション･フィールドにオペレーション･コードを書き込み、オペレーション･コードに続いてオペランドが示されるような命令の場合には、オペランド･フィールドにこのオペランドを書き込む。ＢＴフィールドには命令デコード･発行ユニットから送られてくる分岐タグを書き込む。Ｓフィールドに関しては、無条件分岐命令、即値データのオペランド･スタックへのプッシュ命令あるいはスタック上でコピーを作成することのないオペランド･スタックの操作命令（Java VM における pop，pop2，swap）の場合は実行済みとし、その他の命令の場合は未実行としておく。

【００７９】

ポップ操作を含む命令の場合には、ポップすべき語数と同じ数だけＡＰＳからポップされるＣＲＦのエントリのアドレスを、その順で第１～第４ソース･フィールドに書き込む。プッシュ操作を含む命令の場合には、プッシュすべき語数と同じ数だけＡＰＳにプッシュされるＣＲＦのエントリのアドレスを、その順で第１～第２デスティネーション･フィールドに書き込む。

【００８０】

本実施例においては、スタック上でコピーを作成するようなオペランド･スタックの操作命令の場合には、コピー元となるデータを保持すべく割り付けられているＣＲＦのエントリのアドレスをソース･フィールドに、コピー･データを保持

すべく新たに割り付けられるＣＲＦのエントリのアドレスをデスティネーション・フィールドに、一定の対応関係のもとに書き込む。

【００８１】

命令の種類に応じて、オペランド・スタックに対してポップ／プッシュすべき語数（オペランド・スタックの操作命令の場合には、作成するコピーの語数）は決まっているので、オペレーション・フィールドの内容によって、第１～第４ソース・フィールド及び第１～第２デスティネーション・フィールドのうちのいずれが有効であるかを知ることができる。

【００８２】

命令の内容を、ＩＢに書き込むと同時に、命令の種類に応じて、必要な場合に、適切な機能ユニットの空いているＲＳにも、書き込みの行われるＩＢのエントリのアドレス（ここでは、１サイクル当たり１命令までしか発行しないとしているので、トレイル・ポインタの値と一致する）と共に書き込む。ここで、ＲＳへの書き込みが必要でないのは、即値データのオペランド・スタックへのプッシュ命令、スタック上でコピーを作成することのないオペランド・スタックの操作命令及び無条件分岐命令の場合である。本実施例においては、スタック上でコピーを作成するようなオペランド・スタックの操作命令の場合には、その内容を演算ユニット１の空いているＲＳに書き込むことにする。

【００８３】

エントリ・アドレスがＩＢのソース・フィールドに書き込まれる（ＡＰＳからポップされる）ＣＲＦのエントリの各々のＷＣＦフィールド及びデータ・フィールドが読み出され、ＷＣＦが１の場合、次サイクル以降にエントリ・アドレスとデータがＣＤＢに載せられる。

【００８４】

③ 実行ステージ

命令デコード・発行ステージにおいてあるＲＳに書き込まれた命令に関して、原則として以下のような動作が順次行われる。

【００８５】

・各々のＲＳで、そこに書き込まれているソース・データを保持すべきＣＲＦの

エントリのアドレスとＣＤＢを通じて送られてくるものが比較され、一致すればデータが取り込まれる。本実施例においては、ＲＳに命令の内容が書き込まれるのと同じタイミングでＣＤＢを通じて送られてくるデータも当該ＲＳに取り込まれるものとする。

【００８６】

・必要なソース･データが揃い、機能ユニットの実行部が利用可能であれば、当該ＲＳの内容は実行部に渡され、その実行が開始される。この時点で、当該ＲＳにおける当該命令の保持が解除される。

【００８７】

・デコードの際ＩＢのデスティネーション･フィールドに書き込みが行われる（ＡＰＳにＣＲＦのエントリのアドレスがプッシュされる）ような命令の場合、命令の実行の結果得られたデータをデスティネーションであるＣＲＦのエントリのアドレスと共にＣＤＢに載せる。ＣＲＦにおいては、ＣＤＢを通じて送られてくる内容に基づきデータの書き込みを行い、ＷＣＦフィールドを１に変更する。

【００８８】

・以上のような動作が全て正常に終了すれば、当該ＲＳに書き込まれていたエントリ･アドレスにある（当該命令を保持している）ＩＢのエントリのＳフィールドを正常終了に変更する。

【００８９】

以上は、大部分の命令について当てはまる原則的な動作であるが、本実施例の計算機システムにおいては、命令の種類によっては、以下のような例外的な動作が行われる。

【００９０】

・ＬＳＵのＲＳにおいて、オペランド･スタック上のデータをポップしてアドレス計算を行うようなストア命令（Java VM における iastore, lastore, fastore, dastore, aastore, bastore, castore, sastore）が書き込まれている場合には、ソース･データが全て揃っていなくても、アドレス計算に必要なソース･データが揃った時点でストア･アドレスを計算し、ストア･バッファに書き込む。

【００９１】

・ＬＳＵのＲＳにおいて、ストア命令が書き込まれている場合には、ストア･アドレスとストア･データのストア･バッファへの書き込みが共に終了すれば、当該ＲＳに書き込まれていたエントリ･アドレスにある（当該ストア命令を保持している）ＩＢのエントリのＳフィールドをストア実行可能に変更する。前述したように、実際のストアの実行は完了ステージにおいて行う。

〔００９２〕

・演算ユニット１のＲＳにおいて、スタック上でコピーを作成するようなオペランド･スタックの操作命令が書き込まれている場合には、ソース･データが書き込まれると、そのデータを、デスティネーションとして一定の対応関係のもとに書き込まれているＣＲＦのエントリのアドレスと共にＣＤＢに載せる。それぞれのデスティネーションに関するデータ転送が全て正常に終了すれば、当該ＲＳに書き込まれていたエントリ･アドレスにある（当該命令を保持している）ＩＢのエントリのＳフィールドを正常終了に変更する。

〔００９３〕

以上のように、ＩＢに保持されている未実行の命令は、データ駆動の原理に基づき、実行可能となったものから処理されるので、命令実行順序はout-of-orderになる。また、演算ユニット０／１、分岐ユニット及びロード／ストア･ユニットの各機能ユニットは互いに独立に並行して動作する。

〔００９４〕

ある命令の処理において例外事象が発生した場合には、その情報を、その命令を保持しているＩＢのエントリのＳフィールドに書き込むと共に、命令フェッチ･ユニットに例外ベクタを通知する。

〔００９５〕

④ 完了ステージ

ある命令が完了できるためには、プログラム上の順番でその命令よりも前にある命令が全て完了していなくてはならない。

〔００９６〕

ＩＢのヘッダ･ポインタで示されるエントリにおいて、Ｓフィールドが実行済み／正常終了である、あるいはそうなると、そのエントリに書き込まれている命

令の内容に基づいてＣＰＳ及びＣＲＦを操作し、ヘッダ･ポインタの値に１を加える。

【００９７】

ＣＰＳは、命令がデコード･発行された際のＡＰＳの動作を再現すべく操作される。すなわち、ポップ／プッシュ操作を含む命令の場合には、有効なソース･フィールドの内容と同じものを順にＣＰＳからポップし、有効なデスティネーション･フィールドの内容を順にＣＰＳにプッシュする。スタック上でコピーを作成することのないオペランド･スタックの操作命令の場合には、オペランド･スタックに対して行うべき操作をＣＰＳに対して全く同様に行えばよい。本実施例においては、スタック上でコピーを作成するようなオペランド･スタックの操作命令の場合には、有効なソース･フィールド及び有効なデスティネーション･フィールドを参照して、その命令のデコード･発行の際にＡＰＳに対して行われた操作がＣＰＳにおいて再現される。

【００９８】

本実施例においては、上述のＣＰＳに対する操作に伴い、エントリ･アドレスがＣＰＳからポップされるＣＲＦのエントリでは、ＢＢフィールドを０に変更し、そのエントリ･アドレスをＦＬに登録する。

【００９９】

ＩＢのヘッダ･ポインタで示されるエントリにおいて、ストア命令が書き込まれている場合には、Ｓフィールドがストア実行可能である、あるいはそうなると、ＬＳＵに実際のストアの実行を依頼する。こうすれば、データがプログラム上の順番でストアされることが保証できる。さらに、ＣＰＳ及びＣＲＦに対する操作を上と同様に行い、ヘッダ･ポインタの値に１を加える。

【０１００】

以上のように、ヘッダ･ポインタの値に１が加えられることによって、キューから除外されたＩＢのエントリに保持されていた命令は、完了したことになる。その命令よりも前に発行された命令はすべて完了しているので、命令の完了はin-orderで行われることになる。

【０１０１】

ＩＢのヘッダ･ポインタで示されるエントリにおいて、Ｓフィールドが例外事象発生である、あるいはそうなった場合には、その時点におけるＣＰＳ及びＣＲＦによって、プログラムがin-orderで実行された場合の例外発生時点の状態が構成されるので、正確な例外処理が可能である。例外事象の発生した命令以降に発行された命令を全てキャンセルするには、キャンセルされるべき命令が書き込まれているＩＢのエントリの有効なデスティネーション･フィールドに示されるＣＲＦのエントリの各々に対して、そのＢＢフィールドを０に戻し、そのエントリ･アドレスをＦＬに登録することによって、割り付けを解除し、ヘッダ･ポインタの値に１を加えたものをトレイル･ポインタに書き込むことによって、キャンセルされるべき命令を保持しているＩＢのエントリを全てキューから除外すればよい。

【０１０２】

以上が、本発明実施例の計算機システムの全般的な動作である。

【０１０３】

ついで、具体的な動作例について説明する。いま、本実施例の計算機システムで、以下のようなプログラムを実行することを考えよう。

dload [A]　（変数名[A]に対応する倍精度浮動小数点データのロード）
dload [B]　（変数名[B]に対応する倍精度浮動小数点データのロード）
dadd　　（倍精度浮動小数点データ間の加算）
d2f　　（倍精度浮動小数点データの単精度浮動小数点データへの変換）
fload [T]　（変数名[T]に対応する単精度浮動小数点データのロード）
swap　　（スタック上の最上位の２語を入れ替える）
dup_x1　　（スタックトップの語のコピーを作成し、先頭から３語目に割り込ませる）
fsub　　（単精度浮動小数点データ間の減算）
fdiv　　（単精度浮動小数点データ間の除算）
fstore [X]（スタックトップにある単精度浮動小数点データの変数名[X]に対応する格納域へのストア）

【０１０４】

以上のプログラムは、X=(A+B)/{T-(A+B)} の計算を行うものであるが、ＡとＢのデータが倍精度で与えられ、この間の加算を倍精度のまま実行して、得られた加算データを単精度に変換して、以降は単精度で計算を行う、というものである。

【０１０５】

図６～図１４は、本実施例の計算機システムにおいて、上記プログラムを処理する際の動作をサイクル毎に示した説明図であり、以下ではこの図をもとに詳細な動作を説明する。図６～図１４において、ＣＲＦ６及びＩＢ５の各エントリの構成は、それぞれ図３、図５のものと同じである。図６～図１４で空白となっている箇所は、そのフィールドの内容に留意する必要が無いことを意味する。時系列で各構成要素の内容を示すために、各部の符号の後尾にハイフンと各サイクルに対応する数字を添えている。また、図６～図１４において、ＡＰＳ、ＣＰＳ、ＩＢ及びＣＲＦの各エントリは下から順に０、１、２、～ のようにアドレスが付けられているものとする。

【０１０６】

ＣＤＢは３本のバスで構成されているものとする。レイテンシが２サイクル以下の演算命令は演算ユニット０で、それ以外の演算命令は演算ユニット１で実行されるものとする。

【０１０７】

本動作例においては、説明を簡単にするため、変数データは全てデータ･バッファに保持されており、ＣＲＦとデータ･バッファの間のSpill/Fillの動作は行わないものとする。従って、BP_OF_PS の値は終始０である。

【０１０８】

また、本動作例においては、当初、ＡＰＳ、ＣＰＳ、ＩＢ及びＣＲＦは初期化されており、ＦＬにＣＲＦの全てのエントリのアドレスが順に〈０〉，〈１〉，〈２〉，〈３〉‥‥と書き込まれていて、この順で取り出されるものとする。

【０１０９】

以下に、各サイクルにおける動作を、（Ａ）命令デコード･発行、（Ｂ）実行及び（Ｃ）完了の各ステージに分けて詳細に説明する。

【０１１０】

（１－Ａ）第１サイクルの命令デコード･発行ステージ

命令dload [A]のデコード･発行を行う。倍長語の変数データのオペランド･スタックへのロード命令であるので、ＦＬに登録されているフリーなＣＲＦの２エントリ６(0)、６(1)をそのデータを保持すべく割り付け、そのエントリのアドレス〈０〉，〈１〉をＡＰＳにプッシュし、ＡＰＳは３－１のようになる。

【０１１１】

ＣＲＦの６(0)、６(1)の各エントリにおいては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込み、ＣＲＦは６－１のようになる。ここで、本動作例においては、終始分岐タグとして命令デコード･発行ユニットから０が送られてくるものとする。

【０１１２】

トレイル･ポインタの値は０であるので、ＩＢのエントリ５(0)に上記命令の内容を書き込み、ＩＢは５－１のようになる。この際、ＡＰＳにプッシュされるＣＲＦのエントリのアドレス〈０〉，〈１〉を各々第１、第２デスティネーション･フィールドに書き込んでいる。さらに、トレイル･ポインタの値に１を加え１にする。ここで、本動作例においては、ＩＢのＳフィールドには、命令が未実行であれば０、実行済み／正常終了あるいはストア命令におけるストア実行可能であれば１が書き込まれるものとする。

【０１１３】

ＩＢのエントリ５(0)に書き込まれるものと同じ上記命令の内容を、ＩＢのエントリのアドレス０と共に、ＬＳＵの空いているＲＳ８３１に書き込む。

【０１１４】

（１－Ｂ）第１サイクルの実行ステージ

実行ステージの動作としては何も行われない。

【０１１５】

（１－Ｃ）第１サイクルの完了ステージ

当初のＩＢのヘッダ･ポインタが示すエントリ５(0)において、命令はまだ書き込まれていないため、完了ステージの動作としては何も行われない。

【０１１６】

（２－Ａ）第２サイクルの命令デコード･発行ステージ

命令dload [B]のデコード･発行を行う。倍長語の変数データのオペランド･スタックへのロード命令であるので、ＦＬに登録されているフリーなＣＲＦの２エントリ６(2)、６(3)をそのデータを保持すべく割り付け、そのエントリのアドレス〈２〉，〈３〉をＡＰＳにプッシュし、ＡＰＳは３－２のようになる。

【０１１７】

ＣＲＦの６(2)、６(3)の各エントリにおいては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込み、ＣＲＦは６－２のようになる。

【０１１８】

トレイル･ポインタの値は１であるので、ＩＢのエントリ５(1)に上記命令の内容を書き込み、ＩＢは５－２のようになる。この際、ＡＰＳにプッシュされるＣＲＦのエントリのアドレス〈２〉，〈３〉を各々第１、第２デスティネーション･フィールドに書き込んでいる。さらに、トレイル･ポインタの値に１を加え２にする。

【０１１９】

ＩＢのエントリ５(1)に書き込まれるものと同じ上記命令の内容を、ＩＢのエントリのアドレス１と共に、ＬＳＵの空いているＲＳ８３２に書き込む。

【０１２０】

（２－Ｂ）第２サイクルの実行ステージ

ＬＳＵの実行部はＲＳ８３１から渡されるロード命令を実行する。即ち、データ･バッファにアクセスし、変数Ａの２語のデータを読み出す。

【０１２１】

（２－Ｃ）第２サイクルの完了ステージ

５－１の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(0)において、Ｓフィールドは０であるので、完了ステージの動作としては何も行われない。

【０１２２】

（３－Ａ）第３サイクルの命令デコード･発行ステージ

命令daddのデコード･発行を行う。オペランド･スタックから４語のソース･デ

ータをポップして演算を行い、倍長語の演算結果をプッシュする演算命令であるので、ＡＰＳから〈０〉，〈１〉，〈２〉，〈３〉をポップし、ＦＬに登録されているフリーなＣＲＦの２エントリ６(4)、６(5)を演算結果を保持すべく割り付け、そのエントリのアドレス〈４〉，〈５〉をＡＰＳにプッシュし、ＡＰＳは３－３のようになる。

【０１２３】

ＣＲＦの６(4)、６(5)の各エントリにおいては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込む。

【０１２４】

トレイル･ポインタの値は２であるので、ＩＢのエントリ５(2)に上記命令の内容を書き込む。この際、ＡＰＳからポップされるＣＲＦのエントリのアドレス〈０〉，〈１〉，〈２〉，〈３〉を各々第１～第４ソース･フィールドに、ＡＰＳにプッシュされる〈４〉，〈５〉を各々第１、第２デスティネーション･フィールドに書き込んでいる。さらに、トレイル･ポインタの値に１を加え３にする。

【０１２５】

ＩＢのエントリ５(2)に書き込まれるものと同じ上記命令の内容を、ＩＢのエントリのアドレス２と共に、演算ユニット０の空いているＲＳ８０１に書き込む（daddの演算のレイテンシは２サイクルであるとする）。

【０１２６】

また、６－２の状態にあるＣＲＦの６(0)、６(1)、６(2)、６(3)の各エントリのＷＣＦフィールド及びデータ･フィールドが読み出され、この場合、いずれのエントリもＷＣＦが０であるので、データ転送の必要はない。

【０１２７】

（３－Ｂ）第３サイクルの実行ステージ

ＬＳＵは、データ･バッファから読み出した変数Ａのデータを構成する２語 A_1、A_2を、それぞれデスティネーションであるＣＲＦのエントリのアドレス〈０〉，〈１〉と共に、ＣＤＢに載せる。これに基づき、ＣＲＦにおいては、エントリ６(0)、６(1)にデータの書き込みを行い、ＷＣＦフィールドを１に変更する。また、同じタイミングでＩＢのエントリ５(2)に書き込まれるものと同じ内容が

書き込まれる演算ユニット０のＲＳ８０１においても、ＣＲＦのエントリのアドレス〈０〉，〈１〉に対応するデータの書き込みが行われる。

【０１２８】

以上で、ＩＢのエントリ５(0)に書き込まれている命令の実行が正常に終了するので、次のサイクルにおいて、５(0)のＳフィールドが正常終了を意味する１に変更される。

【０１２９】

以上の動作と並行して、ＬＳＵの実行部はＲＳ８３２から渡されるロード命令を実行する。即ち、データ･バッファにアクセスし、変数Ｂの２語のデータを読み出す。

【０１３０】

（３－Ｃ）第３サイクルの完了ステージ

５－２の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(0)において、Ｓフィールドは０であるので、完了ステージの動作としては何も行われない。

【０１３１】

（４－Ａ）第４サイクルの命令デコード･発行ステージ

命令d2fのデコード･発行を行う。オペランド･スタックから２語のソース･データをポップして変換演算を行い、１語の演算結果をプッシュする演算命令であるので、ＡＰＳから〈４〉，〈５〉をポップし、ＦＬに登録されているフリーなＣＲＦのエントリ６(6)を演算結果を保持すべく割り付け、そのエントリのアドレス〈６〉をＡＰＳにプッシュし、ＡＰＳは３－４のようになる。

【０１３２】

ＣＲＦのエントリ６(6)においては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込む。

【０１３３】

トレイル･ポインタの値は３であるので、ＩＢのエントリ５(3)に上記命令の内容を書き込む。この際、ＡＰＳからポップされるＣＲＦのエントリのアドレス〈４〉，〈５〉を各々第１、第２ソース･フィールドに、ＡＰＳにプッシュされる〈６〉を第１デスティネーション･フィールドに書き込んでいる。さらに、トレイ

ル･ポインタの値に１を加え４にする。

【０１３４】

ＩＢのエントリ５(3)に書き込まれるものと同じ上記命令の内容を、ＩＢのエントリのアドレス３と共に、演算ユニット０の空いているＲＳ８０２に書き込む（d2fの演算のレイテンシは２サイクルであるとする）。

【０１３５】

また、６－３の状態にあるＣＲＦの６(4)、６(5)の各エントリのＷＣＦフィールド及びデータ･フィールドが読み出され、この場合、いずれのエントリもＷＣＦが０であるので、データ転送の必要はない。

【０１３６】

（４－Ｂ）第４サイクルの実行ステージ

ＬＳＵは、データ･バッファから読み出した変数Ｂのデータを構成する２語 B_1、B_2を、それぞれデスティネーションであるＣＲＦのエントリのアドレス＜２＞，＜３＞と共に、ＣＤＢに載せる。これに基づき、ＣＲＦにおいては、エントリ６(2)、６(3)にデータの書き込みを行い、ＷＣＦフィールドを１に変更する。また、ＩＢのエントリ５(2)に書き込まれているものと同じ内容が書き込まれている演算ユニット０のＲＳ８０１においても、ＣＲＦのエントリのアドレス＜２＞，＜３＞に対応するデータの書き込みが行われる。

【０１３７】

以上で、ＩＢのエントリ５(1)に書き込まれている命令の実行が正常に終了するので、次のサイクルにおいて、５(1)のＳフィールドが正常終了を意味する１に変更される。

【０１３８】

（４－Ｃ）第４サイクルの完了ステージ

５－３の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(0)において、Ｓフィールドは０であるので、完了ステージの動作としては何も行われない。

【０１３９】

（５－Ａ）第５サイクルの命令デコード･発行ステージ

命令 fload [T] のデコード･発行を行う。１語の変数データのオペランド･ス

タックへのロード命令であるので、ＦＬに登録されているフリーなＣＲＦのエントリ６(7)をそのデータを保持すべく割り付け、そのエントリのアドレス〈7〉をＡＰＳにプッシュし、ＡＰＳは３－５のようになる。

【０１４０】

ＣＲＦのエントリ６(7)においては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込む。

【０１４１】

トレイル･ポインタの値は４であるので、ＩＢのエントリ５(4)に上記命令の内容を書き込む。この際、ＡＰＳにプッシュされるＣＲＦのエントリのアドレス〈7〉を第１デスティネーション･フィールドに書き込んでいる。さらに、トレイル･ポインタの値に１を加え５にする。

【０１４２】

ＩＢのエントリ５(4)に書き込まれるものと同じ上記命令の内容を、ＩＢのエントリのアドレス４と共に、ＬＳＵの空いているＲＳ８３１に書き込む。

【０１４３】

（５－Ｂ）第５サイクルの実行ステージ

演算命令daddの内容が書き込まれているＲＳ８０１において、必要なソース･データが全て揃ったので、その内容が演算ユニット０の実行部に渡され、演算が開始される。

【０１４４】

（５－Ｃ）第５サイクルの完了ステージ

５－４の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(0)において、Ｓフィールドが１となったので、５(0)の内容に基づいてＣＰＳ（及びＣＲＦ）を操作する。すなわち、ＩＢのエントリ５(0)のデスティネーション･フィールドに書き込まれている〈0〉,〈1〉をＣＰＳにプッシュし、ＣＰＳは４－５のようになる。さらに、ヘッダ･ポインタの値に１を加え１とし、これで、５(0)の命令は完了したことになる。

【０１４５】

（６－Ａ）第６サイクルの命令デコード･発行ステージ

命令swapのデコード･発行を行う。オペランド･スタック上の最上位の２語を入れ替える命令であるので、同様な操作をＡＰＳに対して行い、ＡＰＳは３－６のようになる。

【０１４６】

トレイル･ポインタの値は５であるので、ＩＢのエントリ５(5)に上記命令の内容を書き込む。この際、命令swapは、スタック上でコピーを作成することのないオペランド･スタックの操作命令であるので、Ｓフィールドは実行済みを意味する１とする。さらに、トレイル･ポインタの値に１を加え６にする。

【０１４７】

（６－Ｂ）第６サイクルの実行ステージ

ＬＳＵの実行部はＲＳ８３１から渡されるロード命令を実行する。即ち、データ･バッファにアクセスし、変数Ｔのデータを読み出す。

【０１４８】

（６－Ｃ）第６サイクルの完了ステージ

５－５の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(1)において、Ｓフィールドが１となったので、５(1)の内容に基づいてＣＰＳ（及びＣＲＦ）を操作する。すなわち、ＩＢのエントリ５(1)のデスティネーション･フィールドに書き込まれている〈２〉,〈３〉をＣＰＳにプッシュし、ＣＰＳは４－６のようになる。さらに、ヘッダ･ポインタの値に１を加え２とし、これで、５(1)の命令は完了したことになる。

【０１４９】

（７－Ａ）第７サイクルの命令デコード･発行ステージ

命令dup_x1のデコード･発行を行う。命令dup_x1は、ワード･スタックが、（右方向に成長するものとして）...., word1, word2 のような状態であるとき、これを ...., word2, word1, word2 と変えるような、スタック上で１語のコピーを作成するオペランド･スタックの操作命令であるので、ＦＬに登録されているフリーなＣＲＦのエントリ６(8)をコピー･データを保持すべく割り付け、３－６のように下から〈７〉,〈６〉となっている状態のＡＰＳを３－７のように〈８〉,〈７〉,〈６〉と変える。

【０１５０】

ＣＲＦのエントリ６(8)においては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込む。

【０１５１】

トレイル･ポインタの値は６であるので、ＩＢのエントリ５(6)に上記命令の内容を書き込む。この際、コピー元となるデータを保持すべく割り付けられているＣＲＦのエントリのアドレス〈６〉を第１ソース･フィールドに、コピー･データを保持すべく新たに割り付けられるＣＲＦのエントリのアドレス〈８〉を第１デスティネーション･フィールドに書き込んでいる。さらに、トレイル･ポインタの値に１を加え７にする。

【０１５２】

ＩＢのエントリ５(6)に書き込まれるものと同じ上記命令の内容を、ＩＢのエントリのアドレス６と共に、演算ユニット１の空いているＲＳ８１１に書き込む。

【０１５３】

また、６－６の状態にあるＣＲＦのエントリ６(6)のＷＣＦフィールド及びデータ･フィールドが読み出され、この場合、ＷＣＦが０であるので、データ転送の必要はない。

【０１５４】

（７－Ｂ）第７サイクルの実行ステージ

演算ユニット０は、５(2)の演算命令の実行を終了しており、演算結果を構成する２語(A+B)_1、(A+B)_2を、それぞれデスティネーションであるＣＲＦのエントリのアドレス〈４〉，〈５〉と共に、ＣＤＢに載せる。これに基づき、ＣＲＦにおいては、エントリ６(4)、６(5)にデータの書き込みを行い、ＷＣＦフィールドを１に変更する。また、ＩＢのエントリ５(3)に書き込まれているものと同じ内容が書き込まれている演算ユニット０のＲＳ８０２においても、ＣＲＦのエントリのアドレス〈４〉，〈５〉に対応するデータの書き込みが行われる。

【０１５５】

ＬＳＵは、データ･バッファから読み出した変数Ｔのデータを、デスティネー

ションであるＣＲＦのエントリのアドレス〈７〉と共に、ＣＤＢに載せる。これに基づき、ＣＲＦにおいては、エントリ６(7)にデータの書き込みを行い、ＷＣＦフィールドを１に変更する。

【０１５６】

以上で、ＩＢの５(2)、５(4)の各エントリに書き込まれている命令の実行が共に正常に終了するので、次のサイクルにおいて、５(2)及び５(4)のＳフィールドが正常終了を意味する１に変更される。

【０１５７】

（７－Ｃ）第７サイクルの完了ステージ

５－６の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(2)において、Ｓフィールドは０であるので、完了ステージの動作としては何も行われない。

【０１５８】

（８－Ａ）第８サイクルの命令デコード･発行ステージ

命令fsubのデコード･発行を行う。オペランド･スタックから２語のソース･データをポップして演算を行い、１語の演算結果をプッシュする演算命令であるので、ＡＰＳから〈７〉,〈６〉をポップし、ＦＬに登録されているフリーなＣＲＦのエントリ６(9)を演算結果を保持すべく割り付け、そのエントリのアドレス〈９〉をＡＰＳにプッシュし、ＡＰＳは３－８のようになる。

【０１５９】

ＣＲＦのエントリ６(9)においては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込む。

【０１６０】

トレイル･ポインタの値は７であるので、ＩＢのエントリ５(7)に上記命令の内容を書き込む。この際、ＡＰＳからポップされるＣＲＦのエントリのアドレス〈７〉,〈６〉を各々第１、第２ソース･フィールドに、ＡＰＳにプッシュされる〈９〉を第１デスティネーション･フィールドに書き込んでいる。さらに、トレイル･ポインタの値に１を加え８にする。

【０１６１】

ＩＢのエントリ５(7)に書き込まれるものと同じ上記命令の内容を、ＩＢのエ

ントリのアドレス７と共に、演算ユニット０の空いているＲＳ８０１に書き込む（fsubの演算のレイテンシは２サイクルであるとする）。

【０１６２】

また、６－７の状態にあるＣＲＦの６(7)、６(6)の各エントリのＷＣＦフィールド及びデータ･フィールドが読み出され、この場合、６(7)のＷＣＦが１であるので、次のサイクルにおいて、エントリ･アドレス〈7〉とデータＴがＣＤＢに載せられる。

【０１６３】

（８－Ｂ）第８サイクルの実行ステージ

演算命令d2fの内容が書き込まれているＲＳ８０２において、必要なソース･データが全て揃ったので、その内容が演算ユニット０の実行部に渡され、演算が開始される。

【０１６４】

（８－Ｃ）第８サイクルの完了ステージ

５－７の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(2)において、Ｓフィールドは０であるので、完了ステージの動作としては何も行われない。

【０１６５】

（９－Ａ）第９サイクルの命令デコード･発行ステージ

命令fdivのデコード･発行を行う。オペランド･スタックから２語のソース･データをポップして演算を行い、１語の演算結果をプッシュする演算命令であるので、ＡＰＳから〈８〉，〈９〉をポップし、ＦＬに登録されているフリーなＣＲＦのエントリ６(10)を演算結果を保持すべく割り付け、そのエントリのアドレス〈１０〉をＡＰＳにプッシュし、ＡＰＳは３－９のようになる。

【０１６６】

ＣＲＦのエントリ６(10)においては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込む。

【０１６７】

トレイル･ポインタの値は８であるので、ＩＢのエントリ５(8)に上記命令の内容を書き込む。この際、ＡＰＳからポップされるＣＲＦのエントリのアドレス〈

８〉，〈９〉を各々第１、第２ソース･フィールドに、ＡＰＳにプッシュされる〈１０〉を第１デスティネーション･フィールドに書き込んでいる。さらに、トレイル･ポインタの値に１を加え９にする。

〔０１６８〕

ＩＢのエントリ５(8)に書き込まれるものと同じ上記命令の内容を、ＩＢのエントリのアドレス８と共に、演算ユニット１の空いているＲＳ８１２に書き込む（fdivの演算のレイテンシは１０サイクルであるとする）。

〔０１６９〕

また、６－８の状態にあるＣＲＦの６(8)、６(9)の各エントリのＷＣＦフィールド及びデータ･フィールドが読み出され、この場合、いずれのエントリもＷＣＦが０であるので、データ転送の必要はない。

〔０１７０〕

（９－Ｂ）第９サイクルの実行ステージ

（８－Ａ）で述べたように、エントリ･アドレス〈７〉とデータＴがＣＤＢに載せられ、これに基づき、ＩＢのエントリ５(7)に書き込まれているものと同じ内容が書き込まれている演算ユニット０のＲＳ８０１においても、ＣＲＦのエントリのアドレス〈７〉に対応するデータの書き込みが行われる。

〔０１７１〕

（９－Ｃ）第９サイクルの完了ステージ

５－８の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(2)において、Ｓフィールドが１となったので、５(2)の内容に基づいてＣＰＳ及びＣＲＦを操作する。すなわち、ＩＢのエントリ５(2)のソース･フィールドに書き込まれている〈０〉，〈１〉，〈２〉，〈３〉をＣＰＳからポップし、デスティネーション･フィールドに書き込まれている〈４〉，〈５〉をＣＰＳにプッシュし、ＣＰＳは４－９のようになる。エントリ･アドレスがＣＰＳからポップされるＣＲＦの６(0)、６(1)、６(2)、６(3)の各エントリでは、ＢＢフィールドを０に変更する。ＣＲＦのエントリのアドレス〈０〉，〈１〉，〈２〉，〈３〉をＦＬに登録する。さらに、ヘッダ･ポインタの値に１を加え３とし、これで、５(2)の命令は完了したことになる。

【０１７２】

（１０－Ａ）第１０サイクルの命令デコード･発行ステージ

命令fstore [X]のデコード･発行を行う。スタックトップにある１語のデータのストア命令であるので、ＡＰＳから〈１０〉をポップし、ＡＰＳは３－１０のようになる。

【０１７３】

トレイル･ポインタの値は９であるので、ＩＢのエントリ５(9)に上記命令の内容を書き込む。この際、ＡＰＳからポップされるＣＲＦのエントリのアドレス〈１０〉を第１ソース･フィールドに書き込んでいる。さらに、トレイル･ポインタの値に１を加え１０にする。

【０１７４】

ＩＢのエントリ５(9)に書き込まれるものと同じ上記命令の内容を、ＩＢのエントリのアドレス９と共に、ＬＳＵの空いているＲＳ８３１に書き込む。次のサイクルにおいて、ストア･バッファに変数名[X]に対応するストア･アドレスが書き込まれる。

【０１７５】

また、６－９の状態にあるＣＲＦのエントリ６(10)のＷＣＦフィールド及びデータ･フィールドが読み出され、この場合、ＷＣＦが０であるので、データ転送の必要はない。

【０１７６】

（１０－Ｂ）第１０サイクルの実行ステージ

演算ユニット０は、５(3)の変換演算命令の実行を終了しており、１語の演算結果(A+B)を、デスティネーションであるＣＲＦのエントリのアドレス〈6〉と共に、ＣＤＢに載せる。これに基づき、ＣＲＦにおいては、エントリ６(6)にデータの書き込みを行い、ＷＣＦフィールドを１に変更する。また、ＩＢのエントリ５(6)、５(7)に書き込まれているものとそれぞれ同じ内容が書き込まれている演算ユニット１のＲＳ８１１及び演算ユニット０のＲＳ８０１においても、ＣＲＦのエントリのアドレス〈6〉に対応するデータの書き込みが行われる。

【０１７７】

以上で、ＩＢのエントリ５(3)に書き込まれている命令の実行が正常に終了するので、次のサイクルにおいて、５(3)のＳフィールドが正常終了を意味する１に変更される。

【０１７８】

（１０－Ｃ）第１０サイクルの完了ステージ

５－９の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(3)において、Ｓフィールドは０であるので、完了ステージの動作としては何も行われない。

【０１７９】

以下では、特に記述すべき動作内容のない場合は、実行ステージであれ完了ステージであれ項目を省くことにする。

【０１８０】

（１１－Ｂ）第１１サイクルの実行ステージ

演算命令fsubの内容が書き込まれているＲＳ８０１において、必要なソース･データが全て揃ったので、その内容が演算ユニット０の実行部に渡され、演算が開始される。

【０１８１】

スタック上でコピーを作成するようなオペランド･スタックの操作命令dup_x1の内容が書き込まれているＲＳ８１１において、ソース･データが書き込まれたので、そのデータ(A+B)を、対応関係にあるデスティネーションであるＣＲＦのエントリのアドレス〈8〉と共に、ＣＤＢに載せる。これに基づき、ＣＲＦにおいては、エントリ６(8)にデータの書き込みを行い、ＷＣＦフィールドを１に変更する。また、ＩＢのエントリ５(8)に書き込まれているものと同じ内容が書き込まれている演算ユニット１のＲＳ８１２においても、ＣＲＦのエントリのアドレス〈8〉に対応するデータの書き込みが行われる。

【０１８２】

以上で、ＩＢのエントリ５(6)に書き込まれている命令の実行が正常に終了するので、次のサイクルにおいて、５(6)のＳフィールドが正常終了を意味する１に変更される。

【０１８３】

（１２－Ｃ）第１２サイクルの完了ステージ

５－１１の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(3)において、Ｓフィールドが１となったので、５(3)の内容に基づいてＣＰＳ及びＣＲＦを操作する。すなわち、ＩＢのエントリ５(3)のソース･フィールドに書き込まれている〈４〉，〈５〉をＣＰＳからポップし、デスティネーション･フィールドに書き込まれている〈６〉をＣＰＳにプッシュし、ＣＰＳは４－１２のようになる。エントリ･アドレスがＣＰＳからポップされるＣＲＦの６(4)、６(5)の各エントリでは、ＢＢフィールドを０に変更する。ＣＲＦのエントリのアドレス〈４〉，〈５〉をＦＬに登録する。さらに、ヘッダ･ポインタの値に１を加え４とし、これで、５(3)の命令は完了したことになる。

【０１８４】

（１３－Ｂ）第１３サイクルの実行ステージ

演算ユニット０は、５(7)の演算命令の実行を終了しており、１語の演算結果T-(A+B)を、デスティネーションであるＣＲＦのエントリのアドレス〈９〉と共に、ＣＤＢに載せる。これに基づき、ＣＲＦにおいては、エントリ６(9)にデータの書き込みを行い、ＷＣＦフィールドを１に変更する。また、ＩＢのエントリ５(8)に書き込まれているものと同じ内容が書き込まれている演算ユニット１のＲＳ８１２においても、ＣＲＦのエントリのアドレス〈９〉に対応するデータの書き込みが行われる。

【０１８５】

以上で、ＩＢのエントリ５(7)に書き込まれている命令の実行が正常に終了するので、次のサイクルにおいて、５(7)のＳフィールドが正常終了を意味する１に変更される。

【０１８６】

（１３－Ｃ）第１３サイクルの完了ステージ

５－１２の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(4)において、Ｓフィールドが１であるので、５(4)の内容に基づいてＣＰＳ（及びＣＲＦ）を操作する。すなわち、ＩＢのエントリ５(4)のデスティネーション･フィールドに書き込まれている〈７〉をＣＰＳにプッシュし、ＣＰＳは４－１３のようになる

。さらに、ヘッダ･ポインタの値に１を加え５とし、これで、５(4)の命令は完了したことになる。

〔０１８７〕

（１４－Ｂ）第１４サイクルの実行ステージ

演算命令fdivの内容が書き込まれているＲＳ８１２において、必要なソース･データが全て揃ったので、その内容が演算ユニット１の実行部に渡され、演算が開始される。

〔０１８８〕

（１４－Ｃ）第１４サイクルの完了ステージ

５－１３の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(5)において、Ｓフィールドが１であるので、５(5)の内容に基づいてＣＰＳ（及びＣＲＦ）を操作する。すなわち、（６－Ａ）におけるＡＰＳの動作が再現され、ＣＰＳは４－１４のようになる。さらに、ヘッダ･ポインタの値に１を加え６とし、これで、５(5)の命令は完了したことになる。

〔０１８９〕

（１５－Ｃ）第１５サイクルの完了ステージ

５－１４の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(6)において、Ｓフィールドが１であるので、５(6)の内容に基づいてＣＰＳ（及びＣＲＦ）を操作する。すなわち、（７－Ａ）におけるＡＰＳの動作が再現され、ＣＰＳは４－１５のようになる。さらに、ヘッダ･ポインタの値に１を加え７とし、これで、５(6)の命令は完了したことになる。

〔０１９０〕

（１６－Ｃ）第１６サイクルの完了ステージ

５－１５の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(7)において、Ｓフィールドが１であるので、５(7)の内容に基づいてＣＰＳ及びＣＲＦを操作する。すなわち、ＩＢのエントリ５(7)のソース･フィールドに書き込まれている〈7〉,〈6〉をＣＰＳからポップし、デスティネーション･フィールドに書き込まれている〈9〉をＣＰＳにプッシュし、ＣＰＳは４－１６のようになる。エントリ･アドレスがＣＰＳからポップされるＣＲＦの６(7)、６(6)の各エントリで

は、ＢＢフィールドを０に変更する。ＣＲＦのエントリのアドレス〈7〉，〈6〉をＦＬに登録する。さらに、ヘッダ･ポインタの値に１を加え８とし、これで、５(7)の命令は完了したことになる。

【０１９１】

（２４－Ｂ）第２４サイクルの実行ステージ

演算ユニット１は、５(8)の演算命令の実行を終了しており、１語の演算結果(A+B)/{T-(A+B)}を、デスティネーションであるＣＲＦのエントリのアドレス〈１０〉と共に、ＣＤＢに載せる。これに基づき、ＣＲＦにおいては、エントリ６(10)にデータの書き込みを行い、ＷＣＦフィールドを１に変更する。また、ＩＢのエントリ５(9)に書き込まれているものと同じ内容が書き込まれているＬＳＵのＲＳ８３１においても、ＣＲＦのエントリのアドレス〈１０〉に対応するデータの書き込みが行われる。

【０１９２】

以上で、ＩＢのエントリ５(8)に書き込まれている命令の実行が正常に終了するので、次のサイクルにおいて、５(8)のＳフィールドが正常終了を意味する１に変更される。

【０１９３】

（２５－Ｂ）第２５サイクルの実行ステージ

ストア命令fstoreの内容が書き込まれているＲＳ８３１において、ストア･データが書き込まれたので、そのデータをストア･バッファに書き込む。

【０１９４】

以上で、ＩＢのエントリ５(9)に書き込まれているストア命令に関して、ストア･アドレスとストア･データのストア･バッファへの書き込みが共に終了するので、次のサイクルにおいて、５(9)のＳフィールドがストア実行可能を意味する１に変更される。

【０１９５】

（２６－Ｃ）第２６サイクルの完了ステージ

５－２５の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(8)において、Ｓフィールドが１となったので、５(8)の内容に基づいてＣＰＳ及びＣＲＦを操

作する。すなわち、ＩＢのエントリ５(8)のソース･フィールドに書き込まれている〈８〉，〈９〉をＣＰＳからポップし、デスティネーション･フィールドに書き込まれている〈１０〉をＣＰＳにプッシュし、ＣＰＳは４－２６のようになる。エントリ･アドレスがＣＰＳからポップされるＣＲＦの６(8)、６(9)の各エントリでは、ＢＢフィールドを０に変更する。ＣＲＦのエントリのアドレス〈８〉，〈９〉をＦＬに登録する。さらに、ヘッダ･ポインタの値に１を加え９とし、これで、５(8)の命令は完了したことになる。

【０１９６】

（２７－Ｃ）第２７サイクルの完了ステージ

５－２６の状態にあるＩＢのヘッダ･ポインタが示すエントリ５(9)においては、ストア命令が書き込まれており、Ｓフィールドが１となったので、ＬＳＵにデータ･バッファへのストアの実行を依頼する。さらに、５(9)の内容に基づいてＣＰＳ及びＣＲＦを操作する。すなわち、ＩＢのエントリ５(9)のソース･フィールドに書き込まれている〈１０〉をＣＰＳからポップし、ＣＰＳは４－２７のようになる。エントリ･アドレスがＣＰＳからポップされるＣＲＦのエントリ６(10)では、ＢＢフィールドを０に変更する。ＣＲＦのエントリのアドレス〈１０〉をＦＬに登録する。さらに、ヘッダ･ポインタの値に１を加え１０とし、これで、５(9)の命令は完了したことになる。

【０１９７】

以上で、本実施例の計算機システムにおいて $X=(A+B)/\{T-(A+B)\}$ の計算が完了したことになる。

【０１９８】

本発明の計算機システムにおいては、分岐予測に基づく投機的実行を実現することができる。ＡＰＳ履歴ファイルは、投機的実行を可能にするために具備されるものである。条件分岐命令がデコードされるごとに、ＡＰＳ履歴ファイルの１エントリにＡＰＳの全エントリ及びPP_OF_APSの内容を書き込むようになっている。以下に、本実施例の計算機システムにおいて、分岐予測に基づく投機的実行がどのように行われるかについて説明する。

【０１９９】

前述のように、本実施例の計算機システムにおいては、命令デコード･発行ステージにおいて、命令をデコードして、命令の内容に応じてＡＰＳ及びＣＲＦを操作すると共に、命令の内容をＩＢ及び、必要な場合には、適切な機能ユニットの空いているＲＳに書き込むようになっている。初期状態から命令が流れ始め最初の条件分岐命令がデコードされるまでの間、発行される命令に分岐タグとして０を付し、この分岐タグ０を、命令の内容が書き込まれるＩＢのエントリ（と機能ユニットのＲＳ）のＢＴフィールド、及び、割り付けられるＣＲＦのエントリのＣフィールドに書き込む。

【０２００】

最初の条件分岐命令がデコードされ分岐予測が行われる際に、分岐時点の状態を保存するために、ＡＰＳの全エントリ及びPP_0F_APSの内容をＡＰＳ履歴ファイルのアドレス０のエントリに書き込む。上記の分岐予測に基づいた命令の流れにおいては、分岐タグとして１を付し、ＩＢ（、機能ユニットのＲＳ）及びＣＲＦの設定を行う。

【０２０１】

２つ目の条件分岐命令がデコードされた時に、最初の条件分岐命令が未確定である場合、あるいは確定して予測が当たっていた場合には、ＡＰＳの全エントリ及びPP_0F_APSの内容をＡＰＳ履歴ファイルのアドレス１のエントリに書き込む。２段目の分岐予測に基づいた命令の流れにおいては、分岐タグとして２を付し、ＩＢ（、機能ユニットのＲＳ）及びＣＲＦの設定を行う。

【０２０２】

分岐予測が当たり続ければ以後同様に処理が進み、ＡＰＳ履歴ファイルへの書き込みはアドレス順に行われる。また、ＡＰＳ履歴ファイルのアドレスnのエントリに書き込みが行われてから次に書き込みが行われるまでの間に発行される命令には分岐タグとしてn+1を付すものとする。

【０２０３】

分岐予測が外れた場合には、その条件分岐命令以降に発行された命令に付された分岐タグをもとに、各機能ユニットで実行中あるいはＲＳに保持されている命令をキャンセルし、ＣＲＦにおいてＣフィールドで分岐タグを照合してその一致

する全てのエントリの割り付けを解除し（ＢＢフィールドを０に変更し、エントリ･アドレスをＦＬに登録する）、１Ｂにおいてその条件分岐命令以降にキューに加えられたエントリを除外する（トレイル･ポインタの値をその条件分岐命令が書き込まれているエントリの次のアドレスに書き換える）。さらに、同じエントリ･アドレスにあるＣＰＳのエントリとその内容が一致しないＡＰＳの各エントリ及びPP_OF_APSに、その条件分岐命令がデコードされた際にＡＰＳ履歴ファイルに書き込まれた内容をコピーして、正しい位置の命令からデコード･発行を再開する。

【０２０４】

以上のように、本発明の計算機システムにおいては、ＡＰＳ履歴ファイルを用いることによって、条件分岐命令がデコードされ分岐予測が行われる各々の時点の状態を再構成することができるので、分岐予測に基づく投機的実行が可能である。

【０２０５】

以上では、説明を簡単にするため、１サイクルで同時にデコード･発行／完了できる命令は高々１つまでとして説明してきた。本発明の計算機システムにおいては、同時に複数の命令をデコード･発行／完了できる構成とすることが可能である。すなわち、ＦＬがＦＩＦＯキューの構成となっていれば、割り付けのためにフリーな（空いている）ＣＲＦのエントリのアドレスをＦＬから取り出す順番は決まっており、各命令における何語ポップし何語プッシュするかというようなスタック操作の内容を把握して、同時に複数の命令をデコード･発行することができる。また、命令の完了は、基本的には、ＩＢの書き込みの内容をもとに、命令がデコード･発行された際のＡＰＳの動作を再現すべくＣＰＳを操作し、エントリ･アドレスがＣＰＳからポップされるＣＲＦのエントリの割り付けを解除すればよいので、同時に複数の命令を完了できる構成とすることも可能である。

【０２０６】

同時にデコード･発行／完了できる命令の数を多くするほど、命令デコード･発行ユニットその他の制御回路が複雑になると共に、ＡＰＳ、ＣＰＳ、ＩＢ、ＣＲＦやデータ･バッファを構成する各レジスタ･ファイルのポートの数や演算ユニッ

トの数、さらにＣＤＢを構成するバスの数などの点で、より多量のハードウェアが必要となる。

【０２０７】

本発明の計算機システムにおいては、命令デコード･発行ステージの前段において、同時にデコード･発行する複数の命令の内容を統合した形式に変換するような構成とすることも可能である。あるいは、変換済のコードを命令キャッシュに蓄えておくような構成としてもよい。

【０２０８】

たとえば、１サイクル当り２命令までデコード･発行できるような構成をとる場合、前述のX=(A+B)/{T-(A+B)}を計算するプログラムは図１５の図表に示されるような内容に変換される。図１５の図表の各段には、同時にデコード･発行される２つの命令に基づく、PP_OF_APSの増分、ＡＰＳの操作内容及びＩＢの２エントリに書き込まれるべき命令の内容を示している。ここでは、命令発行前のＡＰＳの内容を ..... s2，s1，s0 （右端がスタックトップ）、ＦＩＦＯキューの構成となっているフリー･リストの内容を（取り出される順に）f1，f2，f3 .... として記述しており、命令発行時にそれぞれ対応するＣＲＦのエントリ･アドレスに置き換えられるようになっている。PP_OF_APSの増分の欄で示されるようにＡＰＳのスタックトップの位置が移動するが、ＡＰＳの操作内容の欄では、この移動後のスタックトップの位置が右端に対応している。また、'NC' は「変化なし(No Change)」を意味する。

【０２０９】

命令セットがスタック型の命令及びレジスタ型の命令を共に含むような、本発明に基づく計算機システムも実現可能である。すなわち、前進ポインタ･スタック及び完了ポインタ･スタックに加えて、各論理レジスタにそれぞれ対応して設けられた各エントリに統合レジスタ･ファイルのエントリ･アドレスが書き込まれるようになっている前進レジスタ･マッピング･テーブル及び完了レジスタ･マッピング･テーブルを具備する構成とし、スタック型の命令に関しては前進／完了ポインタ･スタックを操作し、レジスタ型の命令に関しては前進／完了レジスタ･マッピング･テーブルをアクセスするようにする。この場合、前進ポインタ･スタ

ック履歴ファイルの代わりに、各々のエントリに前進ポインタ･スタック及び前進レジスタ･マッピング･テーブル双方の内容が書き込まれるようになっている前進履歴ファイルを具備する必要がある。

【０２１０】

【発明の効果】

以上のように、本発明の計算機システムは、正確な例外処理を保証しつつ、スタックマシンの機械語で記述されたプログラムを out-of-order で処理するものであるが、複数の機能ユニットによる並列処理やそれらのパイプライン化によって効率的な処理を行うことが可能であるという利点がある。

【０２１１】

また、分岐予測に基づく投機的実行や、１サイクル当り複数命令のデコード･発行／完了の可能な構成とすることにより、さらなる高速化が可能である。

【図面の簡単な説明】

【図１】

本発明の一実施例の計算機システムの基本構成を示すブロック図である。

【図２】

本発明の一実施例における前進ポインタ･スタック（ＡＰＳ）３、完了ポインタ･スタック（ＣＰＳ）４、PP_OF_APS ９０と PP_OF_CPS ９１の２つのプッシュ･ポインタ及びボトム･ポインタ BP_OF_PS ９２の関係を示した説明図である。

【図３】

本発明の一実施例における統合レジスタ･ファイル（ＣＲＦ）６の各々のエントリ６(i)の詳細な構成を示した説明図である。

【図４】

本発明の一実施例における命令バッファ（ＩＢ）５の構成を示した説明図である。

【図５】

本発明の一実施例における命令バッファ（ＩＢ）５の各々のエントリ５(i)の詳細な構成を示した説明図である。

【図６】

図６～図１４は、本発明の一実施例における一動作例の、サイクル毎の前進ポインタ･スタック（ＡＰＳ）３、完了ポインタ･スタック（ＣＰＳ）４、命令バッファ（ＩＢ）５及び統合レジスタ･ファイル（ＣＲＦ）６の内容を具体的に示した説明図である。図６はそのうち第１～第３サイクルの動作内容を示した図である。

【図７】

図６に続く第４～第５サイクルの動作内容を示した説明図である。

【図８】

図６～図７に続く第６～第７サイクルの動作内容を示した説明図である。

【図９】

図６～図８に続く第８～第９サイクルの動作内容を示した説明図である。

【図１０】

図６～図９に続く第１０～第１１サイクルの動作内容を示した説明図である。

【図１１】

図６～図１０に続く第１２～第１３サイクルの動作内容を示した説明図である。

【図１２】

図６～図１１に続く第１４～第１５サイクルの動作内容を示した説明図である。

【図１３】

図６～図１２に続く第１６～第２４サイクルの動作内容を示した説明図である。

【図１４】

図６～図１３に続く第２５～第２７サイクルの動作内容を示した説明図である。

【図１５】

本発明の計算機システムが１サイクル当り２命令までデコードできるような構成をとる場合に、プログラムがどのように変換されるかを具体的に示す図表である。

【符号の説明】

３　前進ポインタ･スタック（ＡＰＳ）

３ａ　前進ポインタ･スタック履歴ファイル（ＡＰＳ履歴ファイル）

４　完了ポインタ･スタック（ＣＰＳ）

５　命令バッファ（ＩＢ）

６　統合レジスタ･ファイル（ＣＲＦ）

７　フリー･リスト（ＦＬ）

８　共通データ･バス（ＣＤＢ）

１０　命令キャッシュ

１１　データ･キャッシュ

１２　データ･バッファ

２０　命令フェッチ･ユニット

２１　命令デコード･発行ユニット

８０　演算ユニット０

８１　演算ユニット１

８２　分岐ユニット

８３　ロード／ストア･ユニット（ＬＳＵ）

９０　PP_OF_APS と名付けたレジスタ

９１　PP_OF_CPS と名付けたレジスタ

９２　BP_OF_PS と名付けたレジスタ

９７　命令バッファ（ＩＢ）のトレイル･ポインタ

９８　命令バッファ（ＩＢ）のヘッダ･ポインタ

８０１、８０２　演算ユニット０のリザベーション･ステーション

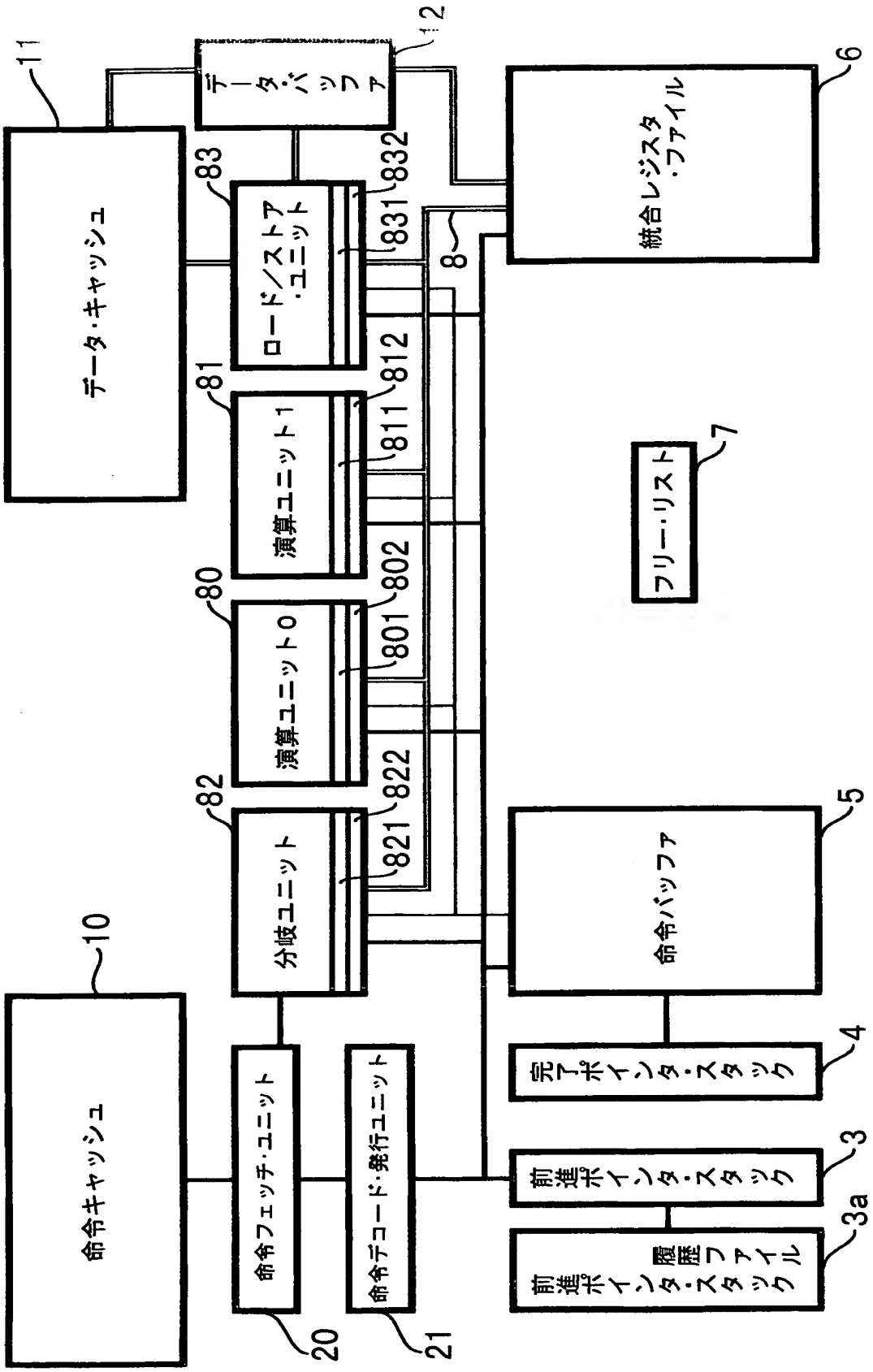
８１１、８１２　演算ユニット１のリザベーション･ステーション

８２１、８２２　分岐ユニットのリザベーション･ステーション

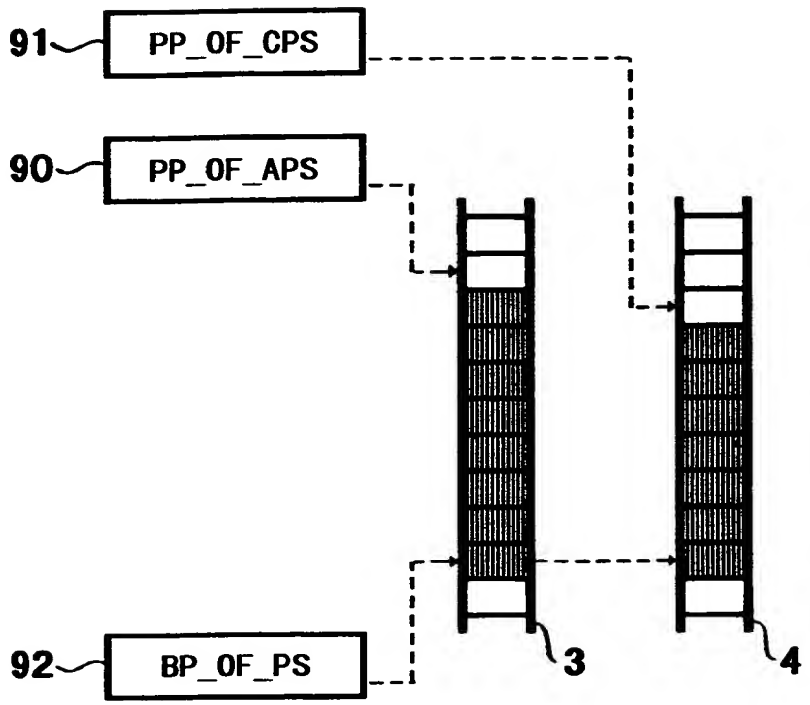
８３１、８３２　ロード／ストア･ユニット（ＬＳＵ）のリザベーション･ステーション

【書類名】　　図面

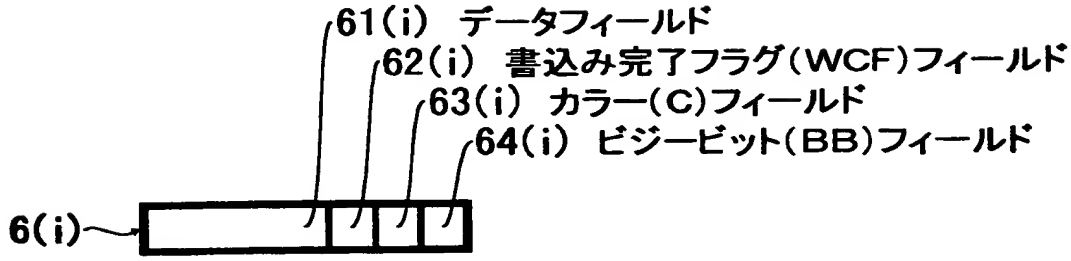
【図１】


データ・キャッシュ
11
データ・バッファ
12
統合レジスタ・ファイル
6
ロード／ストア・ユニット
83
831
832
8
演算ユニット1
81
811
812
フリー・リスト
7
演算ユニット0
80
801
802
分岐ユニット
82
821
822
命令バッファ
5
命令キャッシュ
10
命令フェッチ・ユニット
20
命令デコード・発行ユニット
21
完了ポインタ・スタック
4
前進ポインタ・スタック
3
前進ポインタ履歴ファイル・スタック
3a

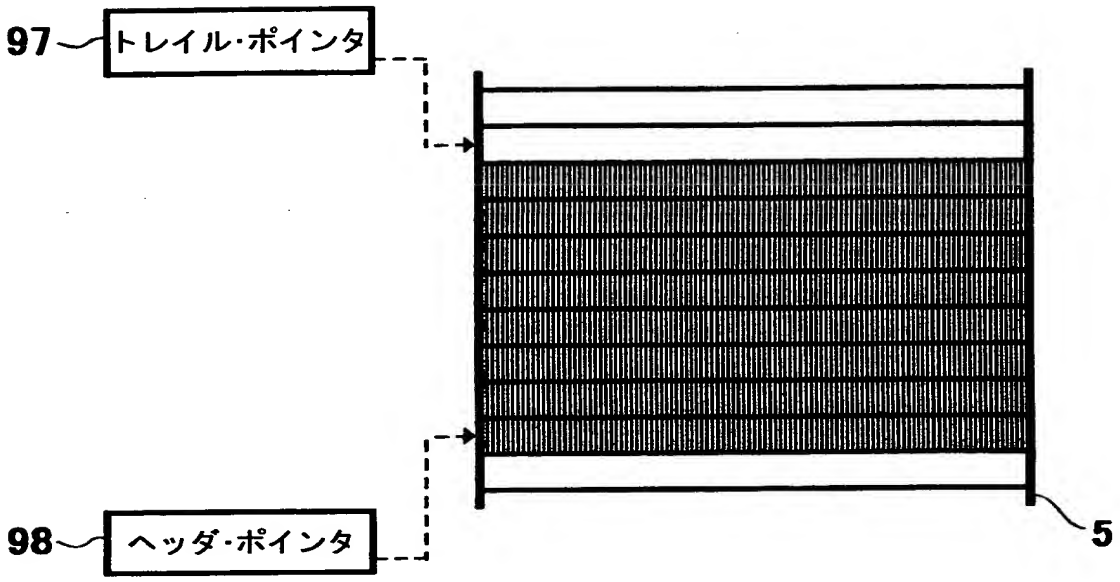
【図２】


91
PP_OF_CPS
90
PP_OF_APS
92
BP_OF_PS
3
4

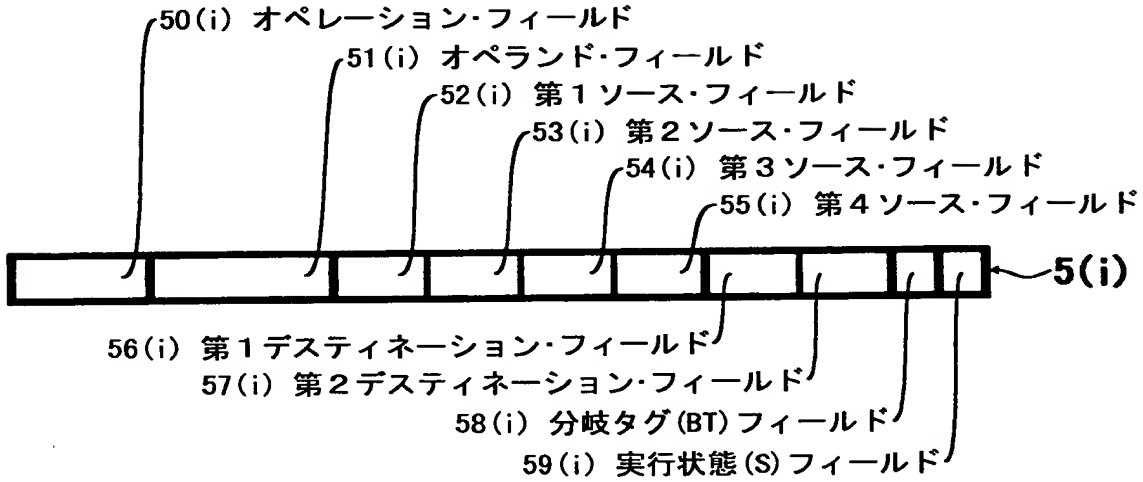
【図３】


61(i) データフィールド
62(i) 書込み完了フラグ(WCF)フィールド
63(i) カラー(C)フィールド
64(i) ビジービット(BB)フィールド
6(i)

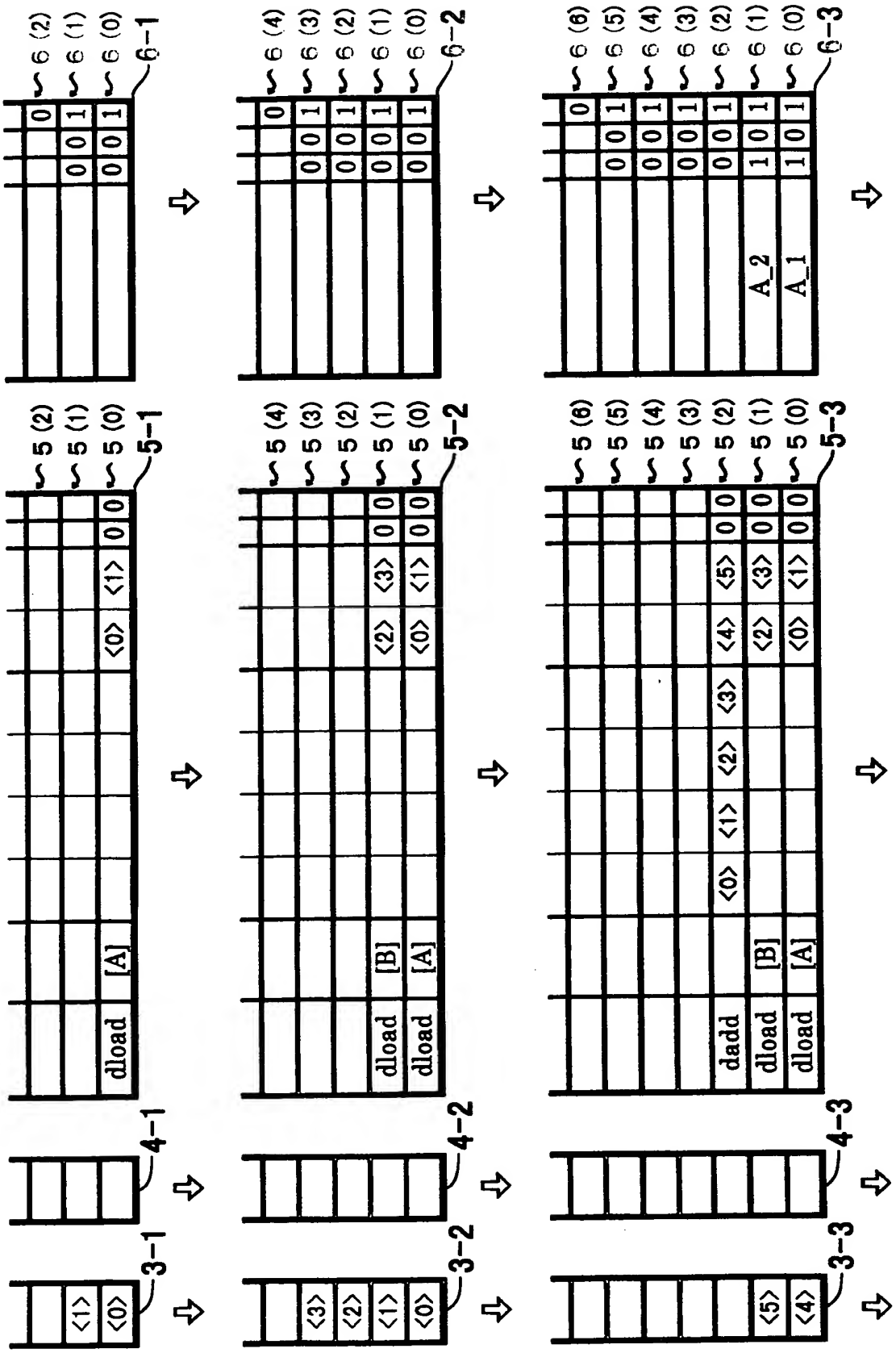
【図４】


97
トレイル･ポインタ
98
ヘッダ･ポインタ
5

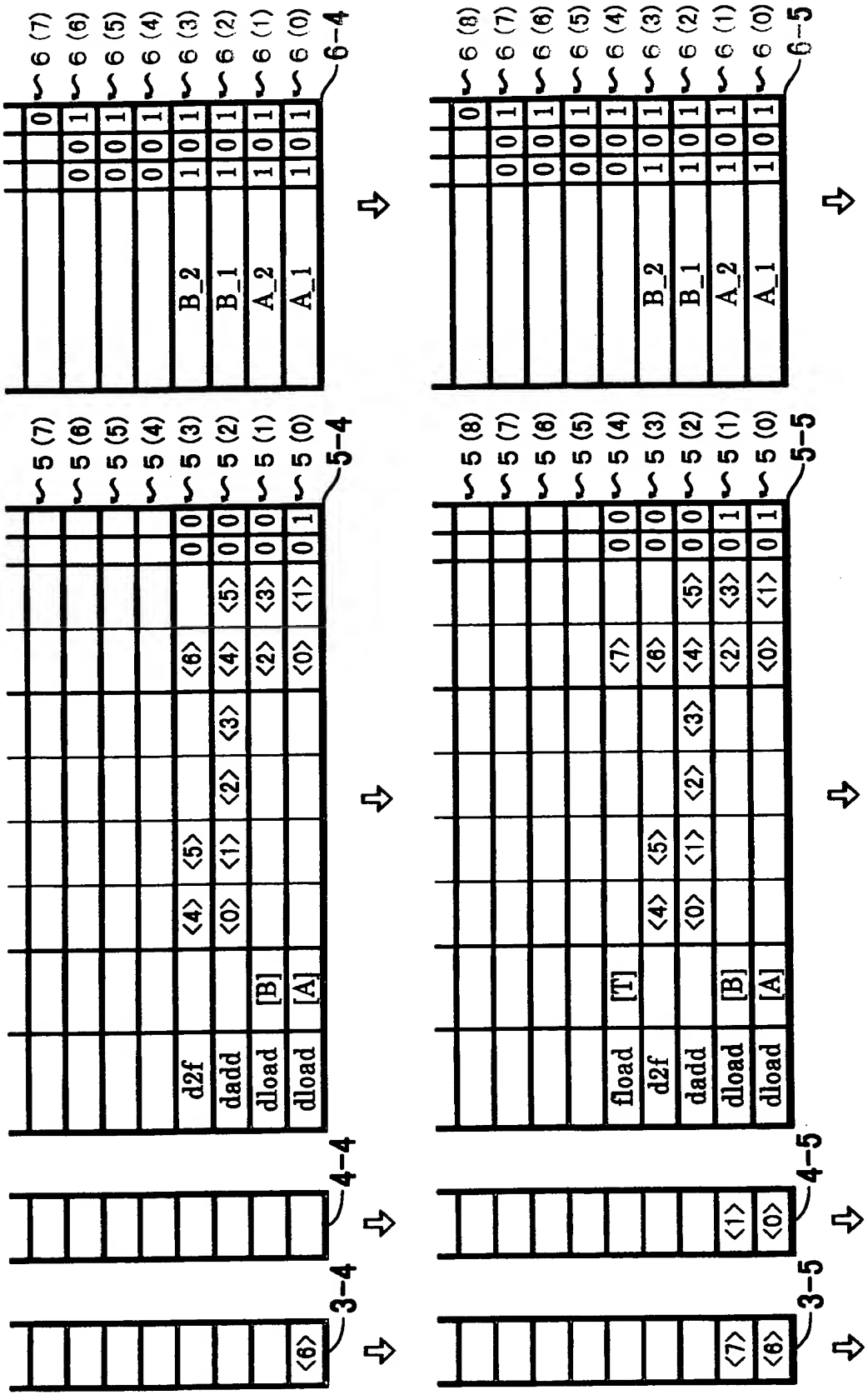
【図５】


50(i) オペレーション･フィールド
51(i) オペランド･フィールド
52(i) 第１ソース･フィールド
53(i) 第２ソース･フィールド
54(i) 第３ソース･フィールド
55(i) 第４ソース･フィールド
5(i)
56(i) 第１デスティネーション･フィールド
57(i) 第２デスティネーション･フィールド
58(i) 分岐タグ(BT)フィールド
59(i) 実行状態(S)フィールド

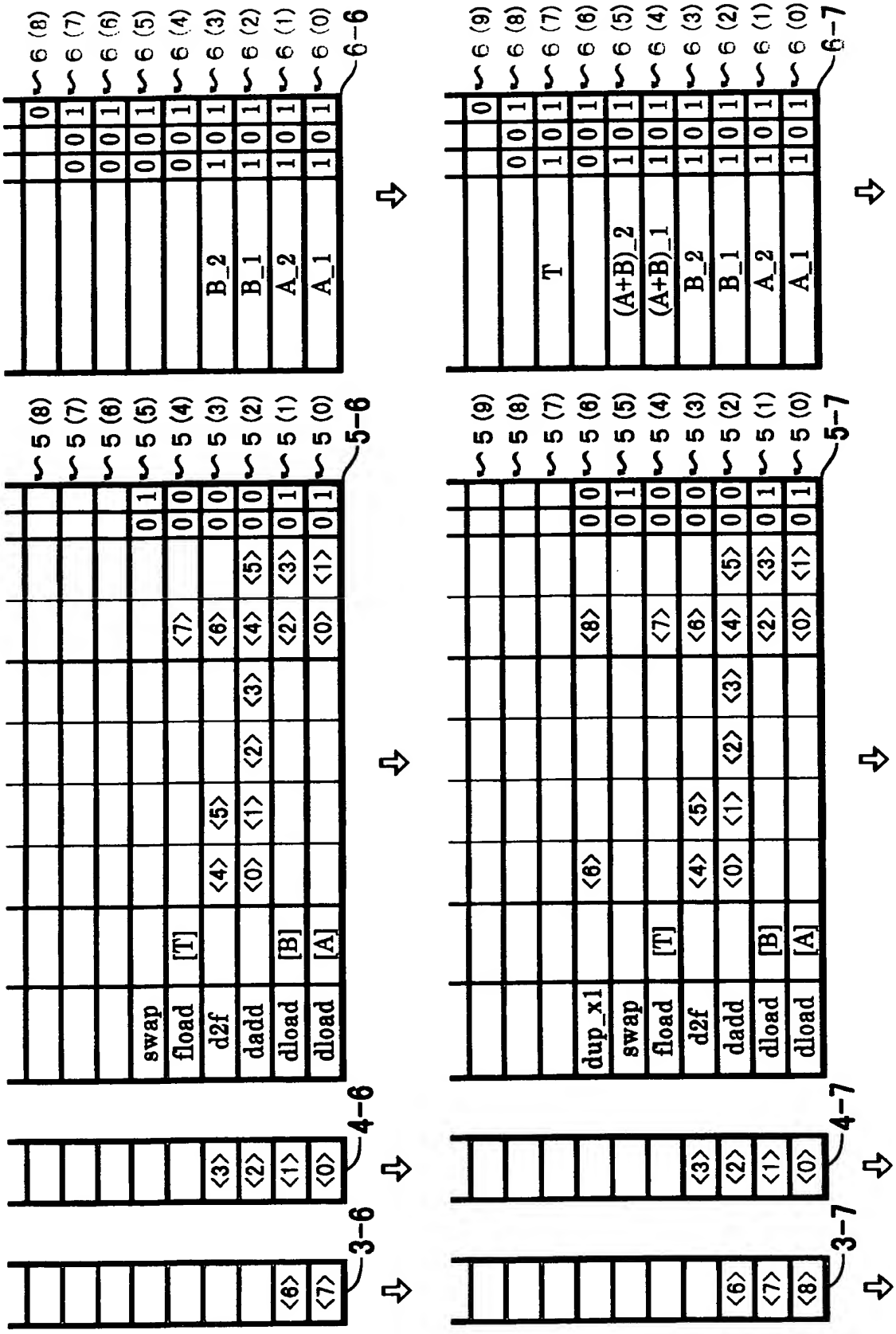
【図６】


6 (2)
6 (1)
6 (0)
6-1
6 (4)
6 (3)
6 (2)
6 (1)
6 (0)
6-2
6 (6)
6 (5)
6 (4)
6 (3)
6 (2)
6 (1)
6 (0)
6-3
A_2
A_1
5 (2)
5 (1)
5 (0)
5-1
5 (4)
5 (3)
5 (2)
5 (1)
5 (0)
5-2
5 (8)
5 (5)
5 (4)
5 (3)
5 (2)
5 (1)
5 (0)
5-3
dload
[A]
[B]
dadd
4-1
4-2
4-3
3-1
3-2
3-3

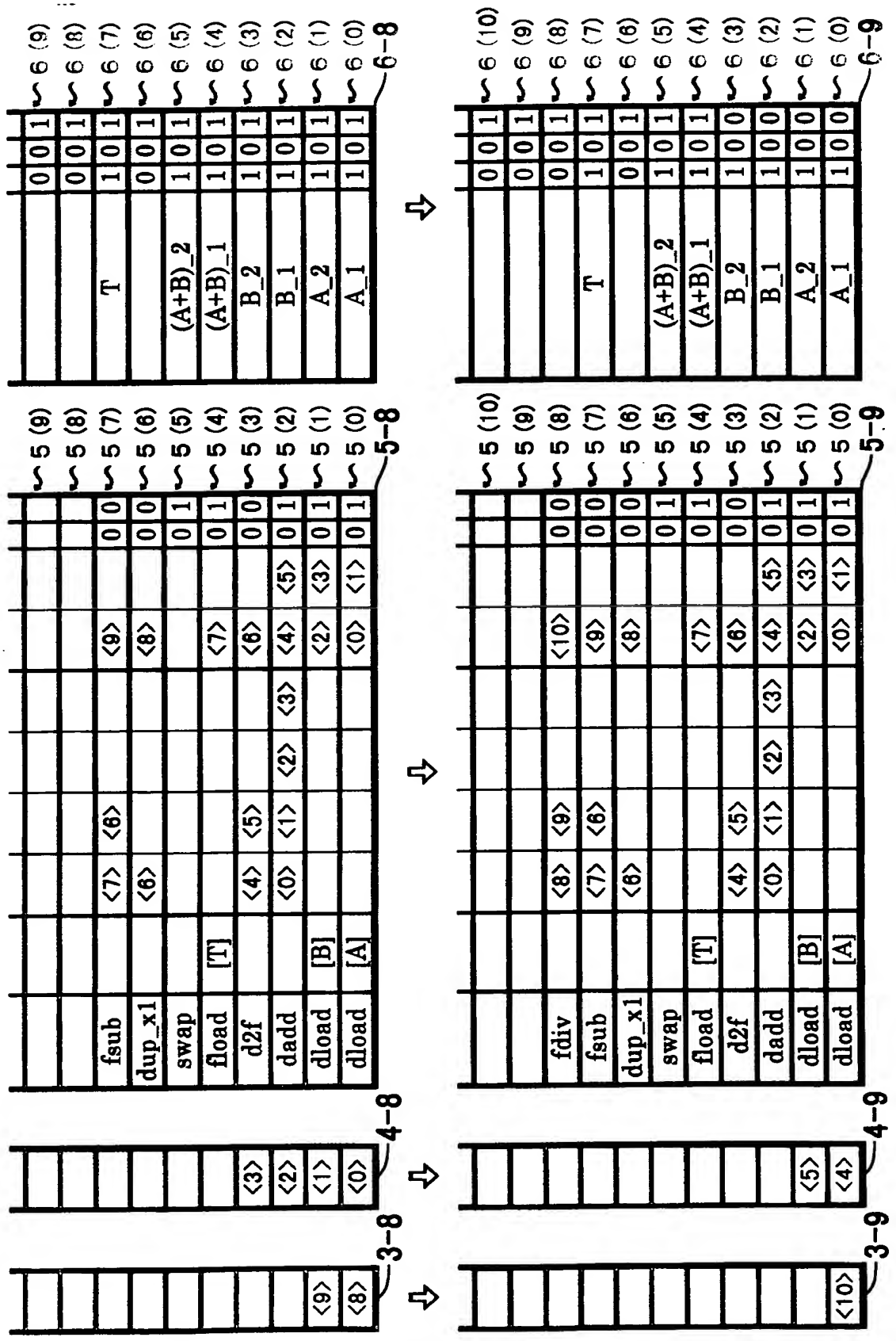
【図７】


6 (7)
6 (6)
6 (5)
6 (4)
6 (3)
6 (2)
6 (1)
6 (0)
6-4
B_2
B_1
A_2
A_1
6 (8)
6-5
5 (7)
5 (6)
5 (5)
5 (4)
5 (3)
5 (2)
5 (1)
5 (0)
5-4
d2f
dadd
dload
dload
[B]
[A]
5 (8)
5-5
fload
[T]
4-4
4-5
3-4
3-5

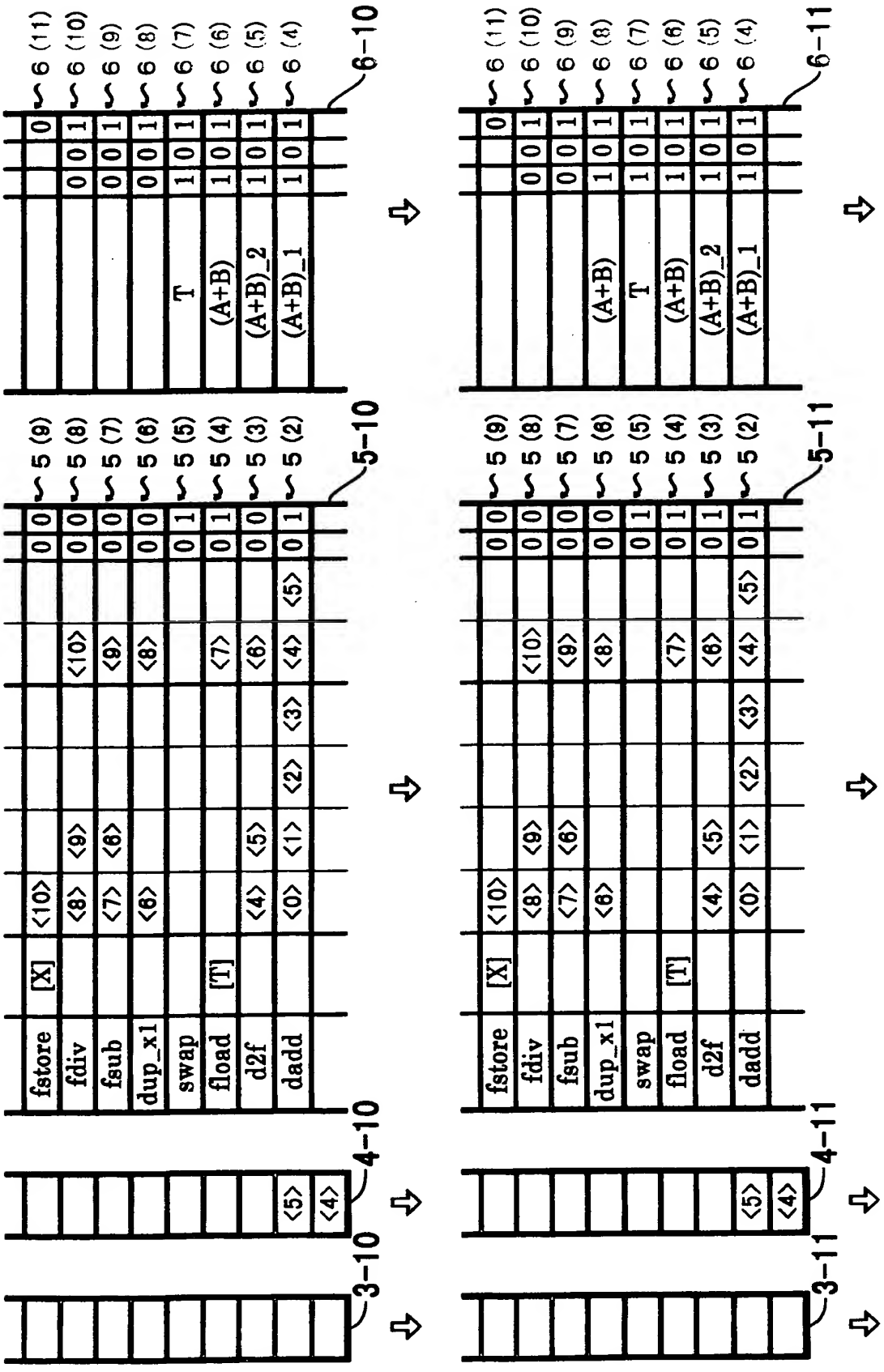
【図８】


6 (8)
6 (7)
6 (6)
6 (5)
6 (4)
6 (3)
6 (2)
6 (1)
6 (0)
6-6
B_2
B_1
A_2
A_1
6 (9)
T
(A+B)_2
(A+B)_1
6-7
5 (8)
5 (7)
5 (6)
5 (5)
5 (4)
5 (3)
5 (2)
5 (1)
5 (0)
5-6
swap
fload
d2f
dadd
dload
[T]
[B]
[A]
5 (9)
dup_x1
5-7
4-6
4-7
3-6
3-7

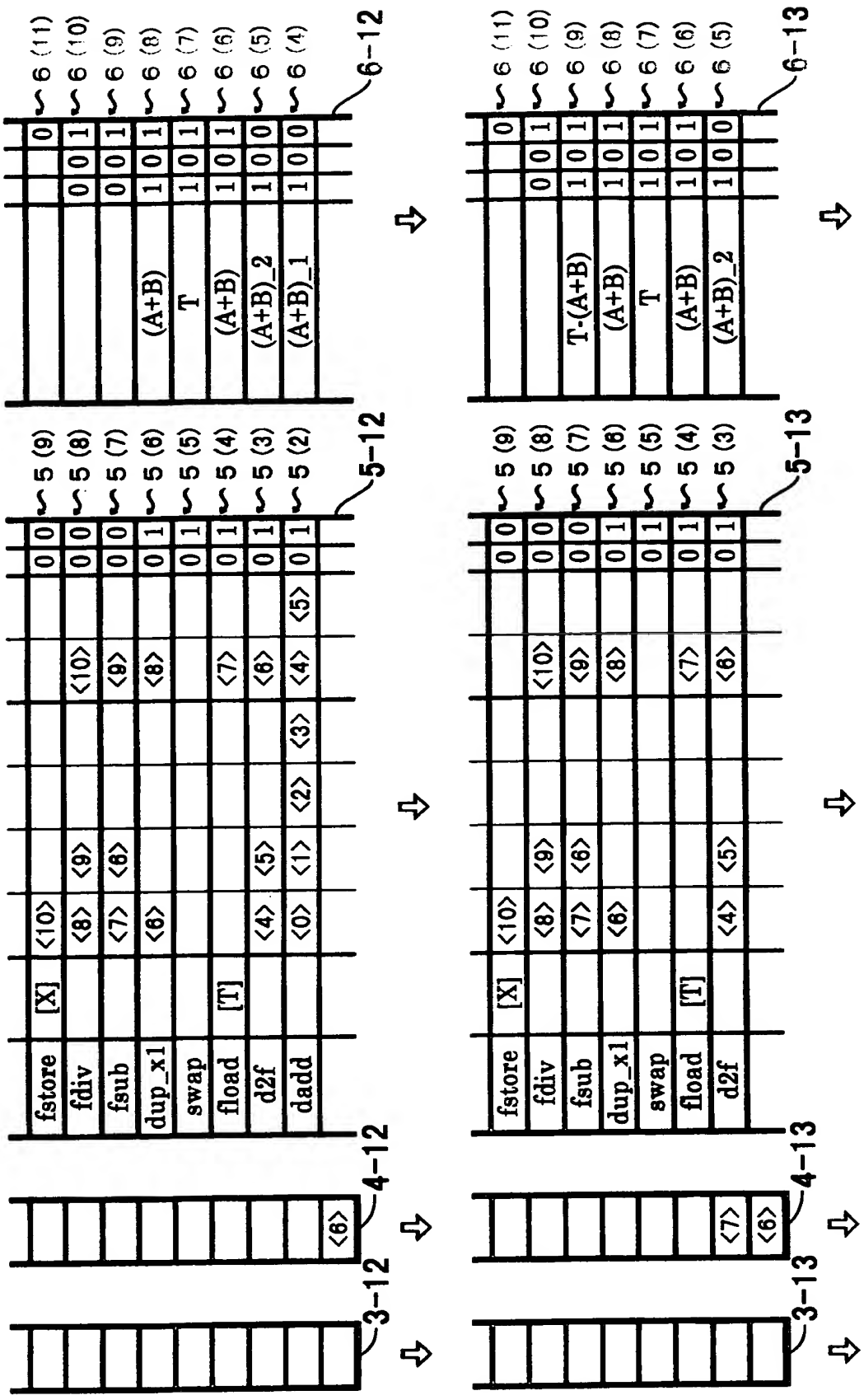
【図９】


6-8
6-9
5-8
5-9
4-8
4-9
3-8
3-9
T
(A+B)_2
(A+B)_1
B_2
B_1
A_2
A_1
fsub
dup_x1
swap
fload
d2f
dadd
dload
[T]
[B]
[A]
fdiv

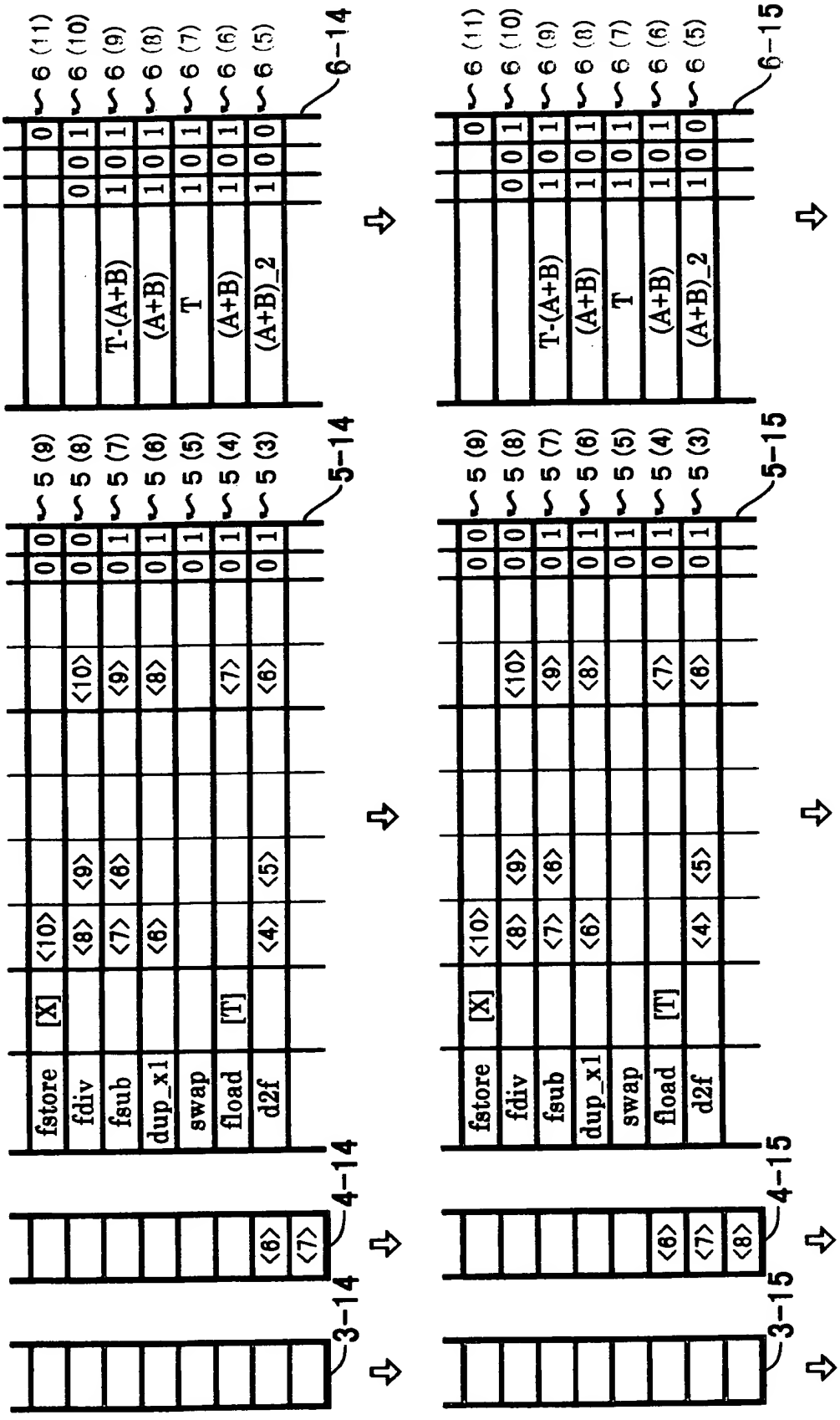
【図１０】


3-10
4-10
5-10
6-10
3-11
4-11
5-11
6-11
fstore
fdiv
fsub
dup_x1
swap
fload
d2f
dadd
[X]
[T]
T
(A+B)
(A+B)_2
(A+B)_1

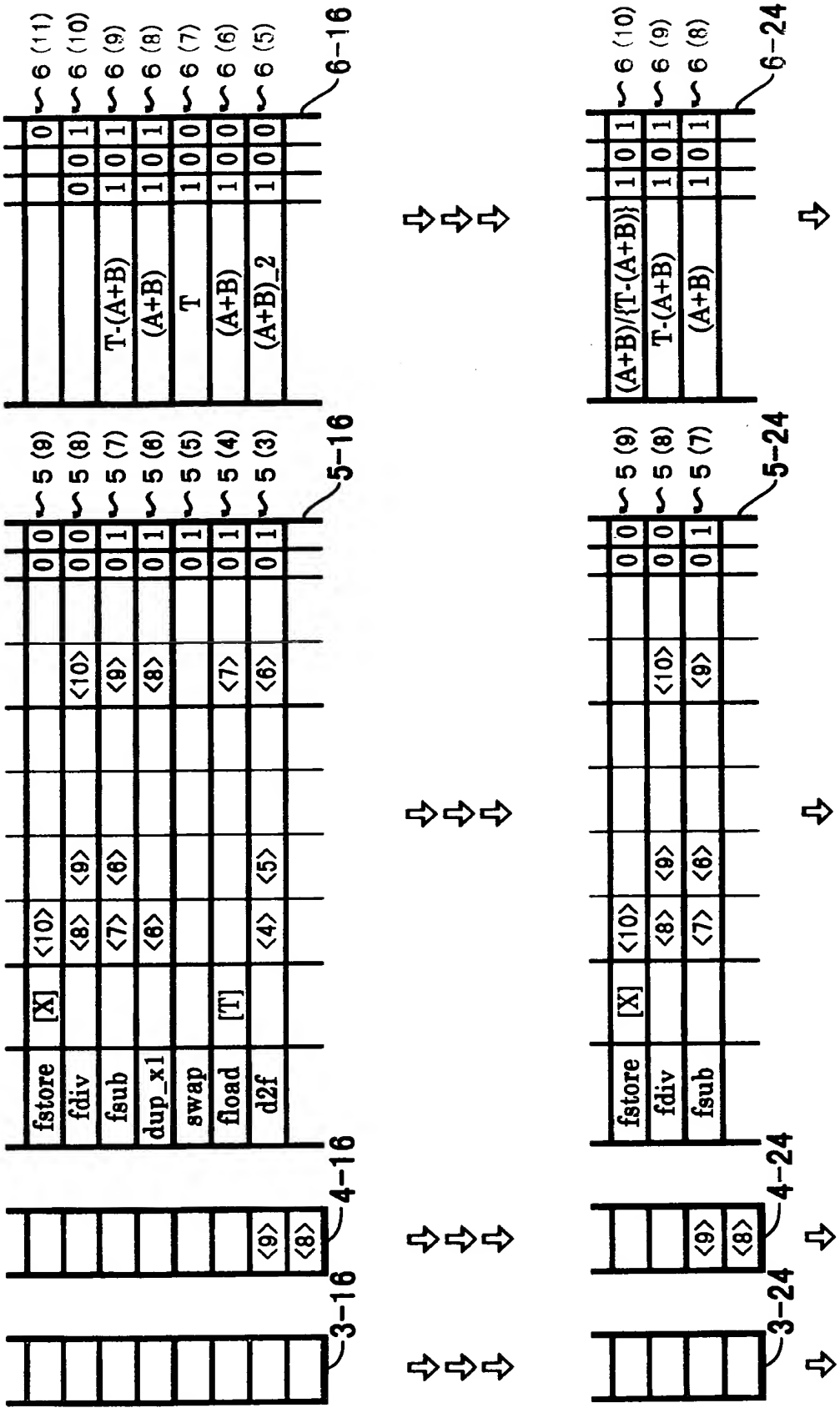
【図１１】


6-12
6-13
5-12
5-13
4-12
4-13
3-12
3-13
fstore
fdiv
fsub
dup_x1
swap
fload
d2f
dadd
[X]
[T]
(A+B)
T
(A+B)_2
(A+B)_1
T-(A+B)

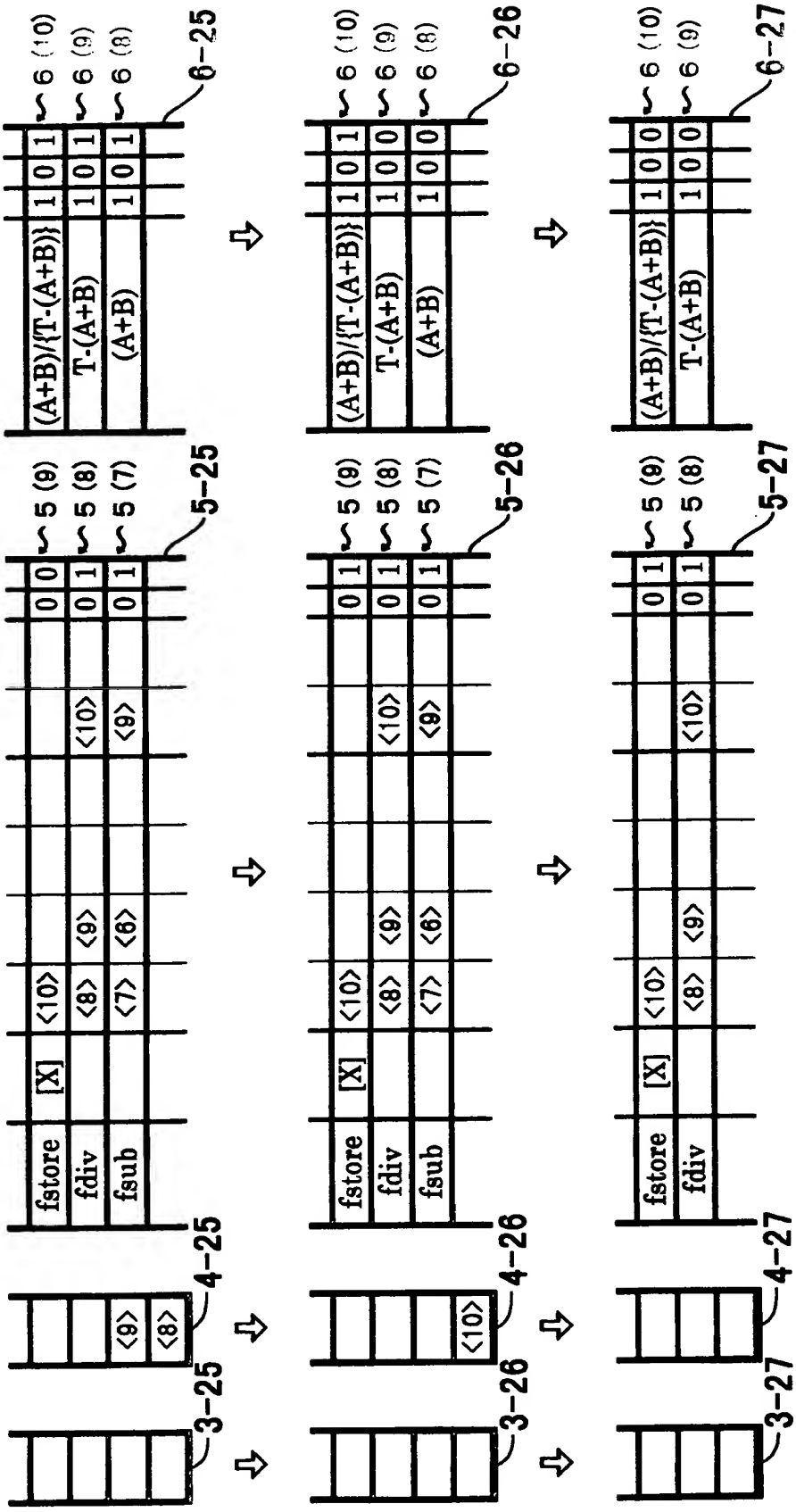
【図１２】


6 (11)
6 (10)
6 (9)
6 (8)
6 (7)
6 (6)
6 (5)
6-14
6-15
T-(A+B)
(A+B)
T
(A+B)
(A+B)_2
5 (9)
5 (8)
5 (7)
5 (6)
5 (5)
5 (4)
5 (3)
5-14
5-15
fstore
fdiv
fsub
dup_x1
swap
fload
d2f
[X]
[T]
4-14
4-15
3-14
3-15

【図１３】


3-16
4-16
5-16
6-16
3-24
4-24
5-24
6-24
fstore
fdiv
fsub
dup_x1
swap
fload
d2f
[X]
[T]
T-(A+B)
(A+B)
T
(A+B)_2
(A+B)/{T-(A+B)}

【図１４】


3-25
4-25
fstore
fdiv
fsub
[X]
<10>
<8>
<9>
<7>
<6>
5-25
5 (9)
5 (8)
5 (7)
6-25
6 (10)
6 (9)
6 (8)
(A+B)/{T-(A+B)}
T-(A+B)
(A+B)
3-26
4-26
5-26
6-26
3-27
4-27
5-27
6-27

【図１５】

PP_OF_APSの増分	ＡＰＳの操作内容	ＩＢへの書き込み内容
+4	NC f1 f2 f3 f4	dload [A] f1 f2 dload [B] f3 f4
−3	NC NC NC NC f3	dadd s3 s2 s1 s0 f1 f2 d2f f1 f2 f3
+1	NC NC NC f1 s0	fload [T] f1 swap
0	NC NC NC f1 f2	dup_x1 s0 f1 fsub s1 s0 f2
−2	NC NC NC NC NC	fdiv s1 s0 f1 fstore [X] f1

【書類名】　　要約書

【要約】

【目的】スタックマシンの機械語で記述されたプログラムを高速で処理する計算機システムを提供する。

【構成】各々のエントリにデータが書き込まれるようになっている統合レジスタ･ファイル６と、各々のエントリに統合レジスタ･ファイルのエントリのアドレスが書き込まれるようになっている前進ポインタ・スタック３と、各々のエントリに個々の命令の内容が書き込まれるようになっている命令バッファ５と、各々適当な数のリザベーション･ステーションを備える機能ユニット群とを具備しており、命令がデコードされるごとに、前進ポインタ・スタック及び統合レジスタ･ファイルを操作すると共に命令の内容を命令バッファ及び、必要な場合には、適切な機能ユニットの空いているリザベーション･ステーションに書き込むことにより、プログラムに含まれる命令がout-of-orderで実行されるべく設定される構成となっている。

【選択図】　　図１

認定・付加情報

特許出願の番号	平成１１年　特許願　第１１５０４７号
受付番号	５９９００３８８４７０
書類名	特許願
担当官	塩崎　博子　　１６０６
作成日	平成１１年　４月２６日

＜認定情報・付加情報＞

【提出日】　平成11年 4月22日

次頁無

出　願　人　履　歴　情　報

識別番号　　〔５９８００３０７０〕

１．変更年月日　　１９９７年１１月２０日
［変更理由］　新規登録
住　所　愛媛県松山市道後喜多町４番３８号
氏　名　関　一